# MARKOV METHODS FOR IDENTIFYING CHIP-SEQ PEAKS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Caitlin Cunningham

January 2013

MARKOV METHODS FOR IDENTIFYING CHIP-SEQ PEAKS

Caitlin Cunningham, Ph.D.

Cornell University 2013

Used to analyze protein interactions with DNA, Chromatin Immunoprecipitation sequencing (ChIP-seq) uses high throughput sequencing technologies to map millions of short DNA "reads" to a reference genome. As the majority of reads map to a protein binding region for a specific protein of interest, a large read count at any given position indicates the presence of a binding region, so that scientists seek "peaks," areas of high counts along the genome. This thesis presents several methods to identify binding regions, utilizing hidden Markov model methods. Unlike existing methods, the final model, HiDe-Peak, accounts for both several major covariates, including mappability and GC content, as well as the dependence between counts present in the dataset. On real data, HiDe-Peak performs in line with existing methods, and in simulations, outperforms its competitors.

## BIOGRAPHICAL SKETCH

Caitlin Cunningham, originally from Needham, Massachusetts, completed her undergraduate degree at the University of Notre Dame, with a double major in Math and History. After two years as an actuary, she pursued a Ph.D. in statistics at Cornell University. She is now an assistant professor at Le Moyne College, in Syracuse, NY.

This thesis is dedicated to my partner, friend, mentor, and supporter, Jonathan.

## ACKNOWLEDGEMENTS

I would like to thank my committee, Dr. Martin Wells and Dr. Giles Hooker. Both were supportive and helpful throughout this process, and I appreciate the time spent thinking about and contributing to my thesis. In addition, I wish to thank the other students from my year in the statistics Ph.D. program, particularly Kirsten Eilertsen, Rajendran Narayanan, and David Zeber. Without their assistance, I would not have succeeded in the first two years of the Ph.D. program. Rajendran in particular provided some assistance with material appearing in chapter 6 of this thesis. I also thank Darcy Steeg Morris and Kelly Kirtland, who, in addition to being great officemates, provided both moral support and minor statistical advice when it was needed. I also thank Anna Bertiger, who, despite not being a statistician, sat through multiple versions of my talks and read and edited research statements and abstracts. Her continued presence and support made much of the work in this thesis possible.

I want to thank the two math professors who most impacted my undergraduate career, Dr. Peter Cholak and Dr. Karen Chandler. I never intended to pursue mathematics after my undergraduate degree, but both encouraged me to follow it further, and their excellent classes helped me to see mathematics as more than just a series of formulas to be memorized.

I need to thank my parents, for their continued support throughout my academic career. They taught all of their children that no goal is too outlandish or impossible, and because of their trust and their belief in us, all of their children have achieved amazing things. They were a bulwark in times of self-doubt, and an amazing cheering section at times of triumph. Similarly, I thank all my siblings and their wives. I could not ask for a more supportive and caring family. I am truly blessed.

The advice, guidance, and direction of Dr. James Booth, my thesis advisor, is visible throughout this thesis, and I am grateful for his continued patience and assistance as we

worked through this process. He has been an invaluable resource, providing insight into both large conceptual problems and small details. He was always generous with his time, and this thesis would not exist without him.

Finally, I must acknowledge my husband, Jonathan Needleman. Jonathan's utter faith in my abilities has been a source of strength for me as I've gone through this process. He's cooked me dinner, taken on more than his share of the chores, and helped keep my life going when I was at my busiest and most stressed. He's held my hand and listened when I needed a listener, and given advice when I needed advice. He's always been interested in my thesis, and has worked hard to understand enough to be truly excited when things are going well. I am so lucky to have had him by my side these last four years, a constant supporting presence. His love is a gift.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

CHAPTER 1

# INTRODUCTION

This thesis develops a methodology to analyze a form of biological data know as ChIP-seq data. ChIP-seq data consists of a long sequence of correlated counts, one count for each position of the human genome, where high counts indicate protein binding regions on the genome. There are a variety of ChIP-seq analysis methods currently available, but many of these ignore specific properties of the data. The method presented in this thesis accounts for the correlated nature of the data, the 'two-strandedness' of the DNA, and has a mechanism to account for both a control and covariates.

## 1.1 Some Biological Background

Within every cell in a living organism is its DNA, short for deoxyribonucleic acid, which contains the genetic instructions for its development and functioning. DNA is made up four nucleotides - Adenine (A), Thymine (T), Guanine (G), and Cytosine (C) - bound in a long double helix. Each position along DNA consists of a 'base-pair', a linked pair of nucleotides, one on each strand of the double helix, with A typically binding to T and G with C. Human DNA is broken up into 22 chromosomes plus two sex chromosomes, X and Y, with two copies (one from each parent) of all the chromosomes except the sex chromosomes. Along the chromosomes are regions called 'genes,' which determine the traits and characteristics of the cell and thus also the organism. Each gene is simply a sequence of the four nucleotides, A, T, G and C, which can be read off by the cell, and which act as instructions to create a specific protein.

The field of genetics is the study of DNA and its genes, specifically the goal of identifying

which genes control which traits. Genes control hair color, eye color, and other physical traits, but can also increase a person's likelihood of developing specific diseases. Lately, there have been many studies attempting to identify genes for a variety of diseases, including cancer, alzheimer's, or obesity. Identifying these genes can be difficult, because these diseases are often caused by many external factors in addition to genetics. An additional complication is that it is often a combination of genes that affect disease outcomes.

As the functions of more genes are being identified, scientists are increasingly looking at 'epigenetics,' which studies the way in which our cells interpret and interact with our DNA. All the cells in an organism share the same DNA, but different cells have mechanisms to read different genes depending on their functions. Hence our skin cells and liver cells behave differently despite having the same exact DNA. Epigenetisists study the mechanisms in cells that cause them to read some genes but not others.

One main mechanism that controls gene expression are proteins called transcription factors. Transcription factors help control cell development, responses to intercellular signals, responses to the environment, and cell cycle control. Transcription factors bind next to genes of interest, and either encourage increased gene expression, or repress gene expression. Knowing where a transcription factor binds, therefore, can give scientists information on which genes are being turned 'on' or 'off' for different cell functions.

Ari Melnick's lab at Cornell's Weill Medical College studies the link between BCL6 and leukemia. BCL6 is a transcription factor known to increase in leukemia cells [2]. If the lab can identify where on the genome BCL6 is binding, then they will be able to identify the genes being affected by the increase in BCL6, and thus those likely linked to leukemia. If the production of BCL6 can be halted or interfered with, then they can prevent cells from manifesting leukemia, even if the genes for leukemia are present in the cell.

The experimental process currently used to identify the binding sites of transcription factors is called Chromatin Immunoprecipitation with high-throughput sequencing (ChIP-seq). ChIP-seq experiments result in sequences of counts millions of positions long, where relatively high values determine binding sites for the protein of interest. The goal of this thesis is to create a statistically sound and computationally feasible method to determine the locations of these 'ChIP-seq peaks,' and to thus determine binding sites.

## 1.2 ChIP-seq

ChIP-Seq is a method used to analyze protein interactions with DNA, in particular a type of protein called transcription factors. Transcription factors bind to specific sequences of DNA, usually next to genes, controlling the expression of that particular gene.

ChIP-Seq consists of two parts: Chromatin Immunoprecipitation (or ChIP) and sequencing (Seq). ChIP works by using living cells (hence, called an *in vivo* procedure), and uses the following steps:

- First, the factor of interest is bound (cross-linked) to the DNA at its binding locations. This cross-linking attaches the protein factor to the DNA, preventing the DNA and protein from separating in future steps of the experiment.
- Second, the DNA is broken into pieces using sonication.
- Next, immunoprecipitation is performed. Immunoprecipitation is a way of a separating a protein out of a solution. Because the protein is attached to its binding DNA, this also gathers the DNA fragments that the protein is bound to out of the full set of DNA fragments.

- Next, the protein-linked DNA is heated to detach the protein, leaving the scientist with segments of DNA that were linked to the protein of interest.

The second part, then, is sequencing all of these DNA segments. The sequencing part is done many different ways (ChIP-chip, CChIP, $Q^2$ChIP). In ChIP-Seq, it is done with high throughput sequencers, a new technology that allows all of the reads to be mapped to a reference genome, unlike older technologies that required experimenters to select regions of interest in advance. These high throughput sequencers use only the first 25 base pairs of each read (called 'tags'), so the sequencing causes the experimenter to lose the information of the actual length of the read. In addition, DNA has many long repetitive sequences. Reads from these regions cannot be mapped to unique locations, and are thus discarded. In typical mammalian experiments, 30-40% of the reads may be discarded, but there have been experiments with more.

The high throughput sequencer has flow cells consisting of multiple lanes, between 6 and 8. Usually each lane is a replicate of the experiment, although one lane is often a control, where no immunoprecipitation is performed. Generally, the data counts are summed across all (non-control) lanes, but it is possible (and likely) for there to be lane effects.

Once the reads are assembled the data set can have several forms. There are two types of reads: "forward reads" and "reverse reads." This is because DNA is two stranded, typically referred to as the "forward" and "reverse" strands. Forward reads match the reference genome exactly and go left to right, whereas the reverse reads are the complement of the mapping genome and read right to left. Some methods keep track only of the mapped positions of the 25 base pair tags outputted by the data set, along with which strand they associate to. Others extend each tag directionally along its strand to the average DNA read length to recreate the reads, and then sum across the two strands. In each case, the data

set consists of a long sequence of counts, the vast majority of which (near 80% or higher) are zeros.

Once the DNA reads have been mapped to the reference genome, the goal is to identify high count positions, and thus, estimate locations where transcription factors were bound to the DNA in the original living cells.

## 1.3 Data Properties

The data has several properties that make it unique and difficult to work with. For one thing, the data set consists of a count for every position of the human genome. The human genome consists of 24 unique chromosomes, which range in length from 57 million (chromosome Y) to 247 million positions (chromosome 1). Even when all positions with a count of zero are removed from the data set, the data files are still very large, consisting of text files over 1.5 gigabytes in size. Naturally, even simple calculations require well written code and a fair amount of time.

The original data provided by the Melnick lab divided the data over 6 experimental lanes. Looking at just one of the lanes in the table below across all 24 chromosomes, one can see that typically 80% of the data set is zero. Note however, that though the mean is low, the maximum count at any given position can be quite high, even in the thousands. Note also that the behavior of the reads does seem to vary by chromosome. Chromosome 24 (Y) has particularly low counts, which makes sense as this data comes from a female, and so should have no reads on Chromosome Y, which only appears in males.

Despite the variation in mean and variance across chromosomes, it is important to esti-

| Chr | Mean | Var | % Zero | Max |
|---|---|---|---|---|
| 1 | 0.26 | 3.83 | 0.81 | 1545 |
| 2 | 0.27 | 0.55 | 0.8 | 105 |
| 3 | 0.32 | 0.70 | 0.77 | 116 |
| 4 | 0.24 | 0.45 | 0.81 | 110 |
| 5 | 0.26 | 0.48 | 0.8 | 84 |
| 6 | 0.28 | 0.94 | 0.79 | 274 |
| 7 | 0.37 | 1.29 | 0.74 | 470 |
| 8 | 0.31 | 0.62 | 0.78 | 106 |
| 9 | 0.24 | 0.54 | 0.83 | 128 |
| 10 | 0.24 | 0.53 | 0.82 | 129 |
| 11 | 0.28 | 0.56 | 0.79 | 88 |
| 12 | 0.27 | 0.61 | 0.8 | 129 |
| 13 | 0.27 | 0.51 | 0.8 | 77 |
| 14 | 0.24 | 0.48 | 0.82 | 113 |
| 15 | 0.23 | 0.49 | 0.83 | 101 |
| 16 | 0.23 | 0.55 | 0.83 | 94 |
| 17 | 0.27 | 0.65 | 0.8 | 83 |
| 18 | 0.28 | 0.59 | 0.79 | 101 |
| 19 | 0.24 | 0.74 | 0.83 | 233 |
| 20 | 0.26 | 0.51 | 0.8 | 79 |
| 21 | 0.20 | 0.40 | 0.84 | 60 |
| 22 | 0.21 | 0.47 | 0.86 | 81 |
| 23 | 0.13 | 0.18 | 0.89 | 67 |
| 24 | 0.04 | 0.05 | 0.97 | 63 |

Table 1.1: BCL6 Data Statistics by Chromosome

mate the entire data set at once, and not on a chromosome-by-chromosome basis, because the experiment is performed on full, whole cells, on not on each chromosome separately. A method that analyzes each chromosome separately has a computational advantage over other methods, but it will over-fit. As an example, a chromosome-by-chromosome method will identify peaks on Chromosome Y, even though all the reads on chromosome Y must be experimental error. A method analyzing the entire data set as a whole will find few if any peaks on chromosome Y.

We want to find an appropriate background distribution for the data, as a place to start.

If we assume that peak areas are those with counts above 3, then a preliminary way to fit a background distribution is to look at only those areas with counts strictly below 3. Naturally, some areas with counts of 1 or 2 are in fact peak regions, so no distribution should fit perfectly, but it allows some initial fitting to be tried.

The first distribution to try fitting is the Poisson distribution. Since all areas with counts above 3 are eliminated, the Poisson distribution is fit by using the percentage of zeros to estimate the rate parameter $\lambda$. The efficacy of the fit can thus be checked by comparing the actual number of 1s in the data set to the number of 1s predicted by a Poisson distribution with that $\lambda$. The ratio of actual/expected would be 1 for a good fit, though, since we have not removed all peak areas with possible counts of 1 (since those are unknown), we would expect something slightly below 1.

Done on a chromosome by chromosome level, the value of $\lambda$ varies from .57 to 1.46 with a mean of .99. The ratio of actual number of 1s to expected ranges from 0.58 to 0.80, with an average of .6, far too low to imply a good Poisson fit. This seems to imply an over-dispersed Poisson, so the next distribution examined is the negative binomial.

Because the negative binomial distribution has two parameters, more than just the average number of zeros is required for estimation. In the end, the two ratios #1/#0 and #2/#1 provide enough information to estimate the distribution without being affected by large counts. The goodness-of-fit is evaluated by checking the number of positions greater than 2 to the number predicted by the negative binomial. The negative binomial predicts too few position greater than 2, and while this is partially expected, as we cannot remove the non-background peak regions, which are unknown and consist of many positions with counts greater than 2, the ratios are typically between 1.5 and 2, which is very high.

Both of these preliminary fits show that the data has too many zeros for a Poisson, and

is even more dispersed than a negative binomial. In the methods described in the literature review in chapter 2, many methods choose to use a Poisson or binomial as a background distribution. These initial tests show that a more complex distribution is required.

### 1.3.1 Lane Effects

When performing the experiment, the biologists place the DNA mixture into a flow cell to be used in the high throughput sequencing machine. The flow cell consists of several 'lanes', so that the data is separated out into these lanes. Typically, one of the lanes is a control. The original BCL6 data set provided by the Melnick lab had each experimental lane separated out, so the data could be examined for a possible 'lane effect.'

The data was simplified such that all positions of one and greater were recorded as a one, creating an indicator that notes the presence of at least one read. In a background model with an assumption that all positions are equally likely to have a read present, a simple bernoulli model should fit. If there is no lane effect, than the sum of these indicators across all 5 lanes should be fit by a binomial. However, if there is a lane effect, then the distribution that will fit the sum across the lanes will not be a binomial. If each lane is produced by its own random bernoulli, then a beta binomial will fit instead.

For chromosome 1, table 1.2 gives the fit Beta-Binomial compared to the actual count frequencies is the data. The very good fit is not surprising as we are fitting 6 categories to a model with 3 parameters. Still, it is a strong indication of a lane effect being present in the data.

No publicly available data set is provided with lane information. As such, though it seems clear that a lane effect should be considered, the rest of this thesis tables the issue. It

| Chr 1 | Actual | Beta-Bin |
|---|---|---|
| 0 | 0.4635 | 0.4592 |
| 1 | 0.3069 | 0.3208 |
| 2 | 0.1629 | 0.1517 |
| 3 | 0.0543 | 0.0533 |
| 4 | 0.0110 | 0.0130 |
| 5 | 0.0010 | 0.0017 |

Table 1.2: Beta-Binomial Fit for Chromosome 1

is intended as future work, and the methods described below can be expanded for data with a lane structure, which will be described in later sections.

CHAPTER 2

# LITERATURE REVIEW

ChIP-sequencing was developed in 2007 by Johnson *et al.* [5] by combining standard ChIP methods with high throughput sequencing. Since then, a large number of peaks detection methods have been developed. ChIP-sequencing data sets have a number of specific issues that must be dealt with when identifying peaks, and each paper deals with these issues differently. The major issues are as follows:

**Tag Extension** The high-throughput sequencing technology records and stores only the first 20-27 base pairs of the DNA sequence fragments (called 'tags'). As these fragments are in fact anywhere from 150 to 300 base pairs long on average, each method must decide whether to extend the tags to their full length and use these full reads in their analysis or just use the shortened 25 base pair tags.

**Read Length** If extending the DNA tags to their full lengths, the methods must determine the appropriate length to extend to. Sonication is used in the experimental process to create the DNA fragments, with different frequencies determining approximate DNA fragment length. The researchers can thus use the mean DNA fragment lengths predicted by the sonication frequency, or they can attempt to approximate the full DNA fragment length using the data available.

**Strandedness** DNA is two stranded, with a forward and reverse strand, with each read in the dataset mapped to one or the other. If the reads are extended to their full length, it is easy to combine the forward and reverse strand reads into a single data set. If the reads are not extended, however, then forward strand peaks will appear upstream (to the left) of the binding sites and reverse strand peaks will appear downstream (to the right) of the binding sites, and this needs to be accounted for.

**Binning** The data sets are naturally very large. One easy way to deal with the size of the data set is to 'bin,' joining together data from consecutive positions. Several of the more complex methods bin or evaluate the data using windows to significantly cut down on the computing time.

**Control** Many of the early data sets did not contain a control lane. As such, there are several earlier methods that have no way to account for control data. Later methods have acknowledged the importance of accounting for control data.

**Poisson/Binomial** Some methods find peaks non-parametrically. Those that do fit a distribution to the data generally use the Poisson or Binomial distribution. Fitting a distribution gives the method more power, but it is generally acknowledged that ChIP-seq data follows neither a Poisson nor a Binomial distribution.

**Density** The experimental methods, under perfects conditions, should create peaks that are approximately gaussian in shape. There are two methods that decide to use this fact in their peak detection methods, using a kernel density estimator to fit either a Normal or t-distribution to the peaks.

**Covariates** Several recent papers have acknowledged the presence of several covariates which can affect peak binding, namely GC content and Mappability. DNA is made of four bases, A, C, T and G, and large numbers of G and C bases can affect read binding. Similarly, large sections of the DNA sequence are not unique, thereby making it impossible to map a read uniquely to those positions. Naturally, read count is dependent on which positions are 'mappable' - that is, can have reads mapped to it. Most papers ignore these issues, but some recent papers present methodologies that take these covariates into account.

**Dependence** Typically, the data set is recorded as a long sequence of counts, where each count is the number of reads overlapping the position. Whether the reads are extended

or not, the reads are many positions long, causing dependence along the sequence of counts. Most methods ignore this dependence or account for it by simply binning, but some methods use more complex methods to account for this known dependence.

## 2.1 Early Papers: 2007

The first paper to develop ChIP-seq as a method was Johnson *et al.* [5] in *Science* in August 2007. Chromatin Immunoprecipitation (ChIP) was a standard methodology, previously used with microchip arrays and other sequencing methods. Johnson was the first to combine ChIP with the new high-throughput sequencing technologies to end up with a map of the whole genome, instead of just pre-specified regions of interest. Johnson ran a control lane, and used a simple thresholding technique to determine peaks. Each peak had to have a five-fold increase of reads over the control and to have a minimum of 13 sequence reads present.

Robertson *et al.* (2007) [13] was published only a few days after the Johnson paper in *Nature Methods* and also claims to have developed ChIP-seq. Robertson does not run a control, and simply uses a cut-off of 11 reads to determine peak regions. Naturally, this cut-off value is dependent on the number of reads produced in the data-set. To determine it, they assumed a Poisson background model, with $\lambda$ set as the average count, and selected a cut-off such that the p-value was less than .001.

Mikkelsen *et al.* (2007) [9], also published in August of 2007, is the first to suggest using a Hidden Markov Model for ChIP-seq analysis. However, this method is never described, and the paper by Richard P. Koche that was promised was in fact never published. The other method to identify peaks in this paper extends the reads to their average read lengths, and then creates a null data set by randomly reassigning the reads. Significant deviation

from this null data set indicates peaks. Note that doing this is equivalent to assuming a Poisson distribution on the background, though the paper does not acknowledge this.

All of these early papers combined the data from the forward and reverse strands and extended the reads to their average length. Surprisingly, one of these papers realized the need for a control, and another fit a HMM to account for the dependence. The peak finding techniques, however, are in general quite simple, and depend on a simple threshold to determine peak regions.

## 2.2 Advancements: 2008-2009

Beginning in 2008, a number of papers were published which attempted to create more complex methods for determining peak regions. These papers focused on the importance of having a control, on determining an accurate average peak length, and using the forward and reverse reads separately. Two of these papers proposed more than one peak finding method. Because of the increased complexity of these papers, we will look at each one separately.

### 2.2.1 Sum, Difference, Normalized Difference, Binomial

Nix *et al.* (2008) [10] present four very simple methods to find peak regions taking control data into account. The authors argue that the presence of control data is vital, as the control for ChIP-seq has shown itself to not be a uniform distribution, as one might hope.

Nix *et al.* use four different peak identification methods. Each method makes use of a sliding window (350 bp) to generate summary scores. Overlapping windows are combined

into candidate binding peaks by merging those that exceed a given threshold.

- The "sum" method uses no input control data, but simply sums the number of reads falling within each window.
- The "difference" method is a subtraction of the sum of the reads in the ChIP data minus the sum of the reads in the input control data for each window.
- The "normalized difference" method takes the difference and divides it by the square root of the sum, an estimate of the standard deviation.
- The final method calculated binomial p-values as follows, with $Y$ be number of data reads and $X$ the number of input reads within a particular region. Given $S = X + Y$, $Y$ is assumed to have a binomial distribution with a probability parameter of 0.5 and number of observations $S$.

On a test data set, the normalized difference and binomial p-value methods outperformed the others, with the normalized difference slightly better in the small data set. The 'sum' method performed the worst, highlighting the need to include control data in ChIP-seq experiments.

### 2.2.2 ChIPseeqer

Though published in 2011, ChIPseeqer [3], developed by Elimento *et al.*, was developed much earlier. It uses a straight forward Poisson p-value to determine peak regions, but does take control data into account.

To determine peaks, ChIPseeqer begins by extending each DNA tag to its full average read length as predicted by sonication. The forward and reverse strands are combined and the

number of reads overlapping each given position is counted. Chip-Seeqer then calculates an expected number of reads per nucleotide as follows: (# of reads $\times$ avg. length of read)/(# nucleotides). The method does identify the nucleotides for which a sequence of 30 reads cannot be mapped uniquely, and removes those from the total number of nucleotides in the denominator, thereby accounting somewhat for mappability. Using this expected average as $\lambda$, a Poisson p-value is then calculated for the count at each nucleotide.

This process is repeated for a control input data set. However, because the number of reads for the control input data set is significantly less than for the experimental data, the method takes the $\log_{10}$ of the p-values for both data sets. It then subtracts off the log-transformed p-values of the control data set from the experimental data set. Finally, a threshold is calculated, and all nucleotides with control-adjusted log-transformed Poisson p-values above that threshold are identified as being part of a ChIP-seq peak.

### 2.2.3 SISSRs

Jothi *et al.* (2008) [6] present a methodology called SISSRs (Site Identification from Short Sequence Reads) to find and identify peak regions on the genome. One aspect of their method that makes it unique is their estimation of the mean DNA length fragment. Consider a read $i$ on the forward DNA strand. In expectation, the binding site on read $i$ will happen at half its length, so that if one knew where the binding site $b$ was, an estimate of the read length would be $2d(i, b)$, where $d(i, b)$ is the distance between the start of read $i$ and the binding site. However, the binding site is unknown.

To estimate the binding site, the read $k$ on the reverse DNA strand closest to the read $i$ downstream is identified. Because the read $k$ is on the reverse strand, it is known to

be downstream of the binding site. However, using this read $k$ as the binding site will overestimate the read length. Instead, they find the read $j$ which is the nearest forward strand read to the binding site upstream of $k$. The binding site is assumed to be evenly between $j$ and $k$. The distance from $i$ to the estimate of the binding site is thus $(d(i,j) + d(j,k)/2)$, so that the estimated read length is twice that.

The method finds these estimated read lengths for all forward reads $i$ for which there exists a $j$ and a $k$, requiring that $d(i,k) \leq 500$. The assumption that $d(i,k) \leq 500$ requires that no reads are longer in length than 500. The read length estimates are then averaged together to find an average DNA fragment length F given by:

$$F = \frac{2}{n}\left(\sum_{i=1^n} d(i,j) + d(j,k)/2\right)$$

where $n$ is the total number of reads $i$. This number $F$ is used to estimate the False Discovery Rate.

Once the average read length is estimated, SISSRs proceeds as follows. Like many other methods, it splits the data into windows of length $w$ (default size 20), though its windows overlap by $w/2$. For each window $i$, the net read count $c_i$ is computed by subtracting the number of reverse reads mapped to window $i$ from the number of forward reads mapped to window $i$. When this value changes from positive to negative, there are now more reads mapping to the reverse strand than the forward strand, implying that the window just passed a protein binding site. The transition point coordinate $t$ is defined as the midpoint between the last seen window with positive $c_i$ and the current window, which has a negative $c_i$. Each of these transition points are a candidate binding site. To be a binding site, it must also pass these other tests:

- The number of forward reads $p$ in the region defined by the coordinates $[t - F, t]$ is at least $E$.

- The number of reverse reads $n$ in the region defined by the coordinate $[t, t + F]$ is at least $E$.
- The number of total reads $p + n$ is at least $R$, which is estimated based on the user-set false discovery rate (FDR) $D$.

These requirements require that there is enough data present to make a conclusion. Usually, $E$ is set to be 2. $R$ is selected to control the estimated FDR rate, assuming a Poisson background distribution.

SISSRs finds many more read sites than Robertson *et al.* and Johnson *et al.*. These extra read sites were confirmed with motif analysis, implying that this method is much more sensitive than earlier methods. Taking advantage of the tendency of reads to appear on both strands symmetrically on either side of a binding site helps make this methods much more powerful.

### 2.2.4 MSP, MTC, WTD

Kharchenko *et al.* (2008) [7], consider three different possible methods, and compare them to the methods of Johnson and Robertson across several data sets. The 25bp tags are not extended to their average length, and so the method looks only at the 'tags', the sequenced end of the DNA reads. Tags that cannot map uniquely are removed from the dataset. However, before a non-unique tag is removed, the sequencing algorithm first checks that, if by changing up to two nucleotides of the sequence, it can map uniquely. Tags that map with some small change called are called mismatches.

Kharchenko *et al.* note that as partial mismatches make up 41-75% of the dataset, it

makes sense to use these sequences in the data set. Naturally, shorter tags with mismatches are more likely to be actual errors than longer ones. The authors perform an analysis to determine the reliability of tags with one, two, and no mismatches for a variety of tag lengths. They recommend using only 25bp length sequences when two mismatches are present, 24 and 25 bp sequences for one mismatch, and as short as 19bp sequences for no mismatches.

The authors additionally note that the background distribution is not uniform. In examining the control data, they note three anomalies. The first is high numbers of tags at a single position much higher than the surrounding counts. These singular peaks typically occur at the same position on both strands. The second is non-uniform, wide (>1000bp) clusters of increased tag counts that occurs on one or both strands. The third anomaly is small clusters of strand-specific tag density resembling peaks, despite being control data.

The first anomaly is easy to eliminate, as the very high singular positions can be identified easily and removed. To adjust for the other background anomalies, they suggest subtracting the rescaled background tag counts prior to determining peaks, as well as requiring a minimum ChIP to control tag ratio.

Kharchenko *et al.* present three methods in addition to Johnson and Robertson's. The first, Window Tag Density (WTD), scores positions based on the strand specific counts upstream and downstream of the examined position. The binding score for each position $i$ in the genome is:

$$S_{wtd}(i) = 2\sqrt{p_U n_D} - (p_D + n_U)$$

where $p_D$ and $p_U$ are the number of tags mapping to a positive strand within a distance of $w$ upstream and downstream of position $i$ respectively. Similarly, $n_D$ and $n_U$ correspond to the number of upstream and downstream tags mapping to the negative strand. Window sizes of 200bp and 400bp were used, and were chosen to include the length of the average

strand. High values of $S_{wtd}(i)$ are used to determine peaks. This method finds positions with many reads upstream on the positive strand and downstream on the negative strand, while penalizing those with reads still downstream on the positive and upstream on the negative. At a binding position, all positive reads in the window should be upstream, and all negative reads in the window should be downstream.

The second, Matching Strand Peaks (MSP), determines local peaks on each strand, and then looks for pairs of positive and negative strand peaks of a comparable size the average read length distance apart. To determine peaks on each strand, they calculate a tag density profile using a Gaussian smoothing kernel with a bandwidth based on the estimated peak width. If positive and negative strand peaks comparable in size are found within the estimated binding region width, then the method determines a binding site to be present.

The third method, Mirror Tag Correlation (MTC), scans the genome to identify positions exhibiting pronounced positive and negative-strand tag patterns that mirror each other. The binding score is calculated as:

$$S_{mtc}(i) = \rho\sqrt{S_{wtd}(i)} + S_{wtd}(i)$$

where $\rho$ is the Pearson linear correlation coefficient between tag vectors $v^+$ and $v^-$, such that $v^+(k)$ is the number of tag positions mapping to the positive strand in position $i + k$, and $v^-(k)$ is the number of tag positions mapping to negative at $i - k$.

For all three methods, peaks within distance $w$ of a larger peak were omitted. For results, the WTD method predicts the most precise binding positions for the NRSF data set. The Johnson method performs the worst. For the other data sets examined, however, the MTC is more precise than the WTD.

### 2.2.5 PICS

PICS, by Zhang *et al.* (2011) [16], uses a Bayesian methodology to fit t-distributions to candidate peaks. They begin by pre-processing the data into candidate regions, each of which has a minimum number of reads. They then use a sliding window and count the number of forward reads in the left half and the number of reverse reads in the right half. They retain any windows that contain at least one forward read and one reverse read. For each chromosome, after merging overlapping windows and removing merged regions with less than two forward or reverse reads, they have a disjoint set of candidate regions, each of which they analyze separately.

They then look at a single candidate region at a time. Let $f_i$ and $r_j$ represent the start of the $i^{th}$ and $j^{th}$ forward and reverse reads in a given region, with $i = 1, \ldots, n_f$ and $j = 1, \ldots, n_r$. Note that the number of forward reads, $n_f$, and the number of reverse reads, $n_r$, will vary by region. Then, they jointly model the starts of the reads with a t-distribution as follows:

$$f_i \sim t_4(\mu - \delta/2, \sigma_f^2) \quad \text{and} \quad r_j \sim t_4(\mu + \delta/2, \sigma_r^2) \tag{2.1}$$

where $\mu$ represents the binding site position, $\delta$ is the difference between the maxima of the forward and reverse distribution, which corresponds to the average DNA fragment size, and $\sigma_f$ and $\sigma_r$ measure the corresponding variability in DNA fragment lengths. To allow for the possibility that the sets of forward and reverse reads in a single candidate region were generated by multiple closely-spaced binding positions, they use mixture models.

PICS uses $t$-distributions because they are similar in shape to normals, but have heavier tails, and better match the data structure. The degree of freedom is fixed at 4 to minimize computation. To accommodate possible biases (e.g. in DNA sonication) that result in asymmetric forward and reverse peaks, they use different variance parameters.

They use a Bayesian approach to take advantage of prior information about $\delta$, the length of the DNA fragments. They also put a common prior on the variance parameters to take into account prior information about the variability of the DNA fragment length within a site and to regularize variance estimates when few reads are available. The priors are as follows:

$$\sigma_f^2, \sigma_r^2 \sim IG_a(\alpha, beta) \text{ and } (\delta|\sigma_f^2, \sigma_r^2) \sim \mathrm{N}(\psi, \rho^{-1}/(\sigma_f^{-2} + \sigma_r^{-2}))$$

where $\psi$ represents their best prior guess about the mean fragment length and $\rho$ controls the spread around this guess. Similarly, $\beta/(\alpha - 1)$ represents the best prior guess about the variance of the DNA fragment length, and $\beta^2/(\alpha-1)^2(\alpha-2)$ controls the spread around this guess. They chose $\alpha = 20, \beta = 40000, \psi = 175, \rho = 1$. This results in a fairly non-informative prior for the DNA fragment length, with a mean of 175 bps and a standard deviation of approximately 50 bps. The parameters are then estimated using an EM algorithm.

This is the first method to develop a way to account for multiple binding sites within a region, and it is the only method to use a Bayesian framework in its peak identification methodology.

### 2.2.6 QuEST

Valouev *et al.* (2008) [14] present QuEST, which, like PICS, uses a kernel density approach to find transcription factor binding sites. Rather than extending reads to their full average length, QuEST performs an analysis on the raw tags, performed on each strand separately. They identify half the average difference between positive peaks and the nearest negative peak to be the "peak shift." Once the peak shift is identified, they shift the reads directionally by the peak shift amount and then join the estimated forward and reverse densities to create

a combined density profile (CDP). They then look for maxima along this joined CDP. QuEST uses control data to calibrate a threshold to control the FDR. It also requires each candidate peak to have a sufficient number of tags as compared to the control data. Finally, they give a kernel density estimation-derived score for each QuEST peak when they list their final choices.

They estimate the density profiles for the forward and reverse reads for any position $i$ as:

$$H_{+;-}(i) = \frac{1}{h} \sum_{j=i-3h}^{i+3h} K((j-i)/h) \times C_{+;-}(j),$$

where $h$ is the kernel density bandwidth (they use h=30bp), $K(x) = \exp(-x^2/2)/(2\pi)^{0.5}$ is the Gaussian kernel density function, and $C_{+;-}(j)$ gives the number of 5' read ends at position $j$ for forward and reverse reads, respectively. These density profiles are un-normalized, for computation convenience. The CDP used in actual peak calling is calculated as:

$$H(i) = H_+(i - \lambda) + H_-(i + \lambda)$$

where $\lambda$ is the estimated peak shift parameter.

The peak shift parameter, $\lambda$ is estimated as follows. For regions with more than 600 tags in a window of 300bps, they calculate forward and reverse profiles and find the local maxima. If the local max is 20-fold greater than the next scoring max, for both the forward and reverse strands, and the number of tags was sufficiently higher than the control, then it was selected. The peak shift parameter was half the average distance between peaks on the negative and positive strand.

Using the CDP, candidate peaks are identified where the QuEST score profile achieves a local max within a 20bp window, provided it was higher than the threshold. A peak is eliminated if the lowest point between it and the next adjacent higher peak is above a selcted threshold to to eliminate enrichment caused by an adjacent higher peak. The

remaining peaks were reported as "calls" if (i) the value of the background CDP was lower than the background CDP threshold, or (ii) the ratio of the ChIP CDP to the background CDP exceeded a specified threshold (the 'rescue ratio.')

### 2.2.7 MACS

MACS, by Zhang *et al.* (2008), [15] is a model based method for identifying ChIP-seq peaks that uses a Poisson distribution. They do not extend tags, but rather shift the forward and reverse reads by a "peak shift" to align the peaks and identify peak regions. They note that the peaks should be bimodal around a protein binding site, and notate the difference as $d$, so that the peak shift should be $d/2$. They then shift all the tags directionally by $d/2$ and sum across the two strands.

They estimate $d$ much as the Valouev *et al.* paper, which also had Johnson as a co-author. They proceed by sliding $2\times$bandwidth windows along the genome to find regions with tags more than $m$fold enriched relative to the random tag genome distribution. It then selects 1000 of these 'high quality' peaks, finds the distance between the positive and reverse peaks, and estimates $d$ to be the average. They find $d$ to be relatively low, at about 126bp or lower.

When a control is available, MACS linearly scales the total control tag count to be the same as the experimental tag count. They also remove reads that appear 'too many' times, in comparison to the sequencing depth, as this can be an error from the amplification process of the sequencing. They then model the tag distribution as a Poisson distribution, and keeps windows with significant tag counts, with the exact highest spot being predicted as the precise binding location.

Control samples often have tag distributions with local fluctuation and biases. MACS adjusts for this by using $\lambda_{\text{local}}$ for the Poisson distribution, which is defined as:

$$\lambda_{\text{local}} = \max(\lambda_{BG}, [\lambda_{1k},]\lambda_{5k}, \lambda_{10k})$$

where $\lambda_{1k}, \lambda_{5k}$, and $\lambda_{10k}$ are $\lambda$ estimated form the 1kb, 5kb, or 10kb window centered at the peak location in the control sample, or the ChIP-Seq sample when a control sample is not available (in which case $\lambda_{1k}$ is not used), and $\lambda_{BG}$ is the overall mean of the control sample. $\lambda_{\text{local}}$ captures the influence of local biases. MACS uses $\lambda_{\text{local}}$ to calculate the $p$-value of each candidate peak and removes potential false positives due to local biases. Candidate peaks with $p$-values below a user-defined threshold are called, and the ration between the ChIP-Seq tag count and $\lambda_{\text{local}}$ is reported as the $fold - enrichment$.

### 2.2.8 PeakSeq

PeakSeq, by Rozowsky *et al.* (2009) [4], is the first method to introduce the concept of 'mappability' as a factor to be considered in peak finding. For each position on the genome, they identify if a sequence of length 30 can map uniquely. They then calculate the proportion of positions in a window for which reads map uniquely. The threshold used to determine peaks is then a function of the mappability score within the window, to account for the fact that less reads can map to a window with few mappable positions.

PeakSeq also develops a methodology to account for the fact that the number of reads in the control is much lower than the number of reads in the sample data. Naively, one might wish to multiply the control counts by a number such that the number of reads in the normalized control matches the sample data set. However, PeakSeq wishes for the number of normalized control reads to match only the number of background, and not peak,

reads in the sample data. They do this by removing reads in peaks determined by a first-pass thresholding technique (where the threshold depends on mappability) from the total number of reads in the sample data, and adjust the control to match this new adjusted number.

For each window, PeakSeq then determines the ratio of the number of mapped reads from the sample to the normalized number of control reads. This number can be used to indicate peaks, and should be dependent on the strength of the peak. In addition, they use the binomial distribution to perform a two-sample test that the number of reads are different for sample data versus the normalized control (rounded up to the nearest integer). They use a Bonferonni-type correction to account for the multiple testing problem.

## 2.3 Cutting Edge: 2010-2011

### 2.3.1 HPeak

HPeak, by Qin *et al.* (2010) [12], presents a method for finding ChIP-Seq peaks using a hidden Markov model approach. HPeak extends each tag directionally from its start position. Because the true length of the reads are generally within a range, such as 175-225 base pairs, the authors "down-weight" the last 50 base pairs, gradually reducing the count at each of those positions from 1 to 0. They then partition the entire genome into small bins of fixed length and simply count the number of reads that fall into each bin. If the read only partially covers the bin, then they give it partial weight, giving it a value less than one when counting.

Next, a two-state hidden Markov model on the read counts for each bin is applied to identify blocks of consecutive enriched bins. The counts within each bin are modeled with the

Generalized Poisson (GP) and the Zero Inflated Poisson (ZIP) distributions. The standard Poisson is not appropriate, as the variance can be as much as 10 times the mean in some data sets. The GP distribution has two parameters, and has probability mass function:

$$P(Y = y|\lambda, \phi) = \left(\frac{\lambda}{1+\phi\lambda}\right)^y \frac{(1+\phi y)^{y-1}}{y!} \exp\left\{\frac{-\lambda(1+\phi y)}{(1+\phi\lambda)}\right\}$$

The Zero Inflated Poisson is used for the background, due to the many bins with a count of zero. It also has two parameters, and its p.m.f. is given by:

$$f(x|\pi, \mu) = \begin{cases} (1-\pi) + \pi e^{-\mu} & \text{if } x = 0 \\ \frac{\pi e^{-\mu}\mu^x}{x!} & \text{if } x > 0 \end{cases}$$

where $\pi$ is the proportion of zeros in the mixture distribution.

To account for control data, the method is run on the control lane, assuming a ZIP distribution for the 'peak' regions in the control. The information from the control data is then used to account for the peaks found in the ChIP data when returning found peaks.

For the hidden Markov model parameter estimation, the method uses a Viterbi algorithm. The method iterates between using method of moments to find parameter estimates for the transition and emission probability distributions, and then using those estimates in the Viterbi algorithm to identify the enriched and non-enriched regions. To begin, enriched and non-enriched regions are identified by using a simple threshold cutoff. After the hidden Markov model algorithm is run, a user-selected posterior probability cut-off is applied to determine the final peak regions.

### 2.3.2 MOSAiCS

MOSAICS, developed by Kuan *et al.* (2011) [11], is the first model to present a method that accounts for GC content. In addition, their model accounts for mappability more thoroughly then Rozowsky, and also include a way to account for a control. Kuan et al. follow Rozowsky et al., and assign each position a 'mappability' score. For a given length $L$ and a given position, they see how often a read of length $L$ starting at that position could also map elsewhere in the genome. The mappability score, $\delta_i$, is 1 if a read of $L$ length maps uniquely to that position and 0 otherwise. Rozowsky et al. show that mappability is highly correlated with transcription start sites, and is associated with high read counts.

Kuan *et al.* note that the reads that overlap any nucleotide $i$ can start at positions between $i - L + 1$ and $i$ for forward reads and between $i + L - 1$ and $i$ for backward reads. They therefore extend the definition of a mappability score as follows:

$$m_i = \sum_{k=i-L+1}^{i+L-1} \delta_k/(2L-1)$$

which is just the average of the mappability scores of the whole region of possible starting positions that overlap position $i$.

The second element of genomic data that appears correlated with read counts is GC content. Recall that the DNA consists of four bases - ATGC. A high GC content is correlated with a higher number of reads. They calculate a GC content score, the percentage of bases for a read of length $L$ that are G or C, and average across the read length in a manner similar to the mappability score.

Kuan *et al.* break the genome into non-overlapping bins of size 200bp. They extend each read up to the full fragment length $L$, where $L$ is between 150-200 bp. For each bin, they then record the number of extended reads that overlap the bin as bin-level counts. They

create a mappability score for a bin, $M_j$, which is just the average of the mappability scores of the positions in bin $j$. Similarly, they average the GC scores for each position in the bin for a bin-wide GC score.

Let $Y_j$ be the read counts and $Z_j$ be an unobserved random variable specifying if bin $j$ comes from enriched (Z=1) or nonenriched (Z=0) population of DNA fragments. Then the model is as follows:

$$Y_j|Z_j = 0 \sim N_j, \quad N_j \sim \text{NegBin}(a, a/\mu_j)$$

$$Y_j|Z_j = 1 \sim N_j + S_j,$$

where $S_j$ represents the signal due to protein binding and $N_j$ measures the effect of GC content and mappability. $S_j$ follows a mixture of Negative Binomials:

$$S_j \sim p_1\text{NegBin}(b_1, c_1) + (1 - p_1)\text{NegBin}(b_2, c_2) + k$$

where $k$ is a constant the represents the minimum observable count in an enriched region. So the distribution of observed counts is a mixture model: $\Pr(Y_j = y) = \pi_0 \Pr(Y_j = y|Z_j = 0) + (1 - \pi_0) \Pr(Y_j = y|Z_j = 1)$, where $\pi_0$ is the proportion on non-enriched bins.

When there is a control (input) lane, it makes sense to have $\mu_j$, the mean of $N_j$, which represents mappability bias and GC content, to depend also on the information from the input. Let $X_j$ be the read count for bin $j$ in the input. They then propose the following model for $\mu_j$:

$$\mu_j = \exp\left\{\beta_0 + \left[\beta_M \log_2(M_j + 1) + \beta'_{GC}\mathbf{S}_p(GC_j) + \beta_{X1}X_j^d\right]\mathbf{I}(X_j \leq s) + \beta_{X2}X_j^d\mathbf{I}(X_j > s)\right\}$$

where $s$ and $d$ are tuning parameters, and $\mathbf{S}_p(GC_j)$ is a spline fit to the GC content.

To estimate the MOSAiCS model, an EM algorithm is used. However, the complexity of the enriched bin distribution means there is no closed form representation for it, and

therefore the M-Step requires time-consuming numerical optimization. The parameters are estimated using a four step process:

**Steps 1 and 2** Estimate the parameters of the nonenriched distribution and proportion of unbound bins under some basic assumptions.

**Steps 3 and 4** Utilize a generalized E-M algorithm to obtain the parameters of the enriched distribution. Use Method of Moments estimators in the M-step to estimate parameters of the enriched distribution.

To simplify the estimation, it is assumed that all bins with 0, 1, or 2 counts are non-enriched.

The details of these estimates can be found in the paper. The authors claim the estimation is very fast, and that their algorithm is a significant improvement over FindPeaks, by Rozowsky *et al.*

## 2.4 Overall

The papers presented are classified in table 2.1 in regards to the nine issues mentioned at the beginning of this chapter:

**Tag Extend** The method extends the tags to the average read length.

**Read Length** The method estimates the read length, as opposed to using the sonication value, or, similarly, calculates a 'peak shift' value.

**Strandedness** The method analyses the two strands separately.

**Binning** The method bins the data.

**Control** The method has a way to account for the control.

**Poisson/Binomial** The method uses a Poisson or Binomial distribution for the background.

**Density** The method uses a kernel density estimator on peaks.

**Covariates** The method develops a way to account for covariates like GC content or Mappability.

**HMM** The method uses a Hidden Markov Model to account for dependence.

The final row of the table includes the method presented in this thesis, called "HiDe-Peak."

| Author (Method) | Tag Extend | Read Length | Strandedness | Binning | Control | Parametric | Density | Covariates | HMM |
|---|---|---|---|---|---|---|---|---|---|
| Johnson | X | | | X | X | | | | |
| Robertson | X | | | | | X | | | |
| Mikkelsen | X | | | | | X | | | X |
| Nix (Sum, Difference) | X | | | X | X | X | | | |
| Elimento (ChIPseeqer) | X | | | X | X | X | | | |
| Jothi (SISSRs) | | X | X | | | X | | | |
| Kharchenko (MSP, MTC) | | X | X | | | | | | |
| Zhang, X (PICS) | | X | X | X | X | | X | | |
| Valouev (Quest) | | X | | | | | X | | |
| Zhang, Y (MACS) | | X | | | X | X | | | |
| Rozowsky (PeakSeq) | X | | | X | X | X | | X | |
| Qin (HPeak) | X | | | X | X | | | | X |
| Kuan (Mosaics) | X | | | X | X | | | X | |
| Cunningham (HiDe-Peak) | | | X | X | X | X | | X | X |

Table 2.1: Paper Overview

CHAPTER 3

# MARKOV CHAIN METHODS

The original data set we worked with was ChIP-seq data for BCL6, provided by Dr. Ari Melnick's lab at Weill Medical College. The data extended each short tag to the full DNA read length, and then combined the forward and reverse read counts into a single long sequence of counts. The number of DNA reads present at any position is highly dependent on the positions surrounding it, as each DNA read is as many as 200 positions long. The vast majority of the methods (with the exception of HPeak [12]) explored in chapter 2 ignore this dependence, and in fact assume independence between read counts at consecutive positions.

We wanted to develop a model that accounted for this dependence. For simplicity's sake, we assume each position depends only on the position immediately to its left, so that we can develop a simple Markov chain model for the data. As the Melnick BCL6 data has no control lane associated with it, we needed to develop a null model for the data.

## 3.1 Null Model

We make the following assumption: in the absence of any binding sites, reads are randomly assigned to the genome, such that each position has an equal probability of having a specific read present. For the following null model development, we assume the average length of a read is 250 base pairs.

Let $N$ be the total number of reads mapped to the non-repeating regions of the chromosome. In addition, let $L$ be the number of base pairs in the non-repeating regions of the chromosome. My assumption above defines a null model that states that reads cover any

specific base pair with probability $250/L$, as though the reads have been randomly dropped across the chromosome. This leaves out many of the subtleties of the ChIP-seq process, but seems a reasonable place to start. In addition, we assume that all reads run from left to right, so that when we speak of a read's "starting position", we are referring to its leftmost end.

For a single base pair, say $P_1$, the probability of any read over-lapping that position is simply $250/L$, as there are 250 possible starting positions for that read that will result in it overlapping $P_1$. The count at position $P_1$ is thus binomial:

$$P(\text{count at } P_1 = x_1) = \binom{N}{x_1} \left(\frac{250}{L}\right)^{x_1} \left(1 - \frac{250}{L}\right)^{N-x_1}$$

We then want to know:

$$P(\text{count at } P_2 = x_2 | \text{count at } P_1 = x_1).$$

We have two possibilities then - either $x_2$ is greater than $x_1$ or $x_2$ is less than $x_1$ (the equality case is contained in either of these). Because $L$ is typically over a hundred million in length, we will ignore the cases of the first 250 and last 250 base pairs, where the below derivation has some additional complications.

For $x_2 > x_1$, we have:

- If $x_2$ is greater than $x_1$, then we need at least $x_2 - x_1$ reads to start at position $P_2$. Each of those has the probability $(1/(L - 250))$. We use $L - 250$ in the denominator to account for the fact that none of the reads can overlap $P_1$.
- If exactly $x_2 - x_1$ reads to start at position $P_2$, then $x_1$ reads overlapping $P_1$ must also overlap $P_2$. Each of the events has probability $(249/250)$, as there are 250 starting positions that overlap $P_1$ and 249 of them also overlap $P_2$.

- Finally, the remaining $N - x2$ reads cannot overlap $P_1$ and $P_2$. Each of those have probability $(1 - 1/(L - 250))$

So we can think of the total $N - x_1$ unknown reads as:

$$\begin{aligned} P(\text{count at } P_2 = x_2 | \text{count at } P_1 = x_1) &= \binom{N - x_1}{x_2 - x_1} \left(\frac{1}{L - 250}\right)^{x_2 - x_1} \\ & \left(\frac{249}{250}\right)^{x_1} \left(1 - \frac{1}{L - 250}\right)^{N - x_2} \end{aligned}$$

However, it is possible that some number $j$ of the $x_1$ reads that we assumed covered both $P_1$ and $P_2$ in fact ended at $P_1$ and the exact same number of reads began at $P_2$. We can sum over all possible values of $j$. In addition, this will affect the number of reads that begins at $P_2$ by $j$. Note that this is now just the sum of a product of two binomials:

$$\begin{aligned} P(x_2|x_1) &= \sum_{j=0}^{x_1} \binom{x_1}{j} \left(\frac{249}{250}\right)^{x_1 - j} \left(\frac{1}{250}\right)^{j} \\ & \binom{N - x_1}{x_2 - x_1 + j} \left(\frac{1}{L - 250}\right)^{x_2 - x_1 + j} \left(1 - \frac{1}{L - 250}\right)^{N - x_2 - j} \end{aligned}$$

The case for $x_2 < x_1$ is similar, resulting in:

$$\begin{aligned} P(x_2|x_1) &= \sum_{j=0}^{x_2} \binom{x_1}{x_1 - x_2 + j} \left(\frac{1}{250}\right)^{x_1 - x_2 + j} \left(\frac{249}{250}\right)^{x_2 - j} \\ & \binom{N - x_1}{j} \left(\frac{1}{l - 250}\right)^{j} \left(1 - \frac{1}{(l - 250)}\right)^{N - x_1 - j} \end{aligned}$$

These two formulas are obviously quite similar. We can write them as a unified formula as follows:

$$P(x_2|x_1) = \sum_{j=max(0, x_1 - x_2)}^{x_1} \mathrm{B}\left(j, x_1, \frac{1}{250}\right) \mathrm{B}\left(x_2 - x_1 + j, N - x_1, \frac{1}{L - 250}\right)$$

We can also note that the dependence between two positions $k$ apart is simply:

$$P(x_{i+k}|x_i) = \begin{cases} \sum_{j=max(0,x_1-x_2)}^{x_1} \mathrm{B}\left(j, x_1, \frac{k}{250}\right) \mathrm{B}\left(x_2 - x_1 + j, N - x_1, \frac{k}{L-250}\right) & \text{if } k < 250 \\ \mathrm{B}\left(x_{i+k}, N - x_i, \frac{250}{L-250}\right) & \text{if } k \geq 250 \end{cases}$$

## 3.2 Markov Chain Methods

Once we have this null model, we can consider estimating the probability of a window of size $w$ as simply $P(x_1)\prod_{i=2}^{w} P(x_i|x_{i-1})$. Then, we can select those windows with particularly low null probabilities (as determined by some cutoff) as those with unusual peaks in the data. The challenge to this window-based idea is the selection of a cutoff.

The window joint probabilities can be approximated by using the marginal binomial for the first count, and the Markov probabilities for each succeeding count in the window. One way to select a cut-off for "unusual" data is to find it through simulation. Null data can be simulated, and the joint probability for each possible window (one starting at each base pair) can be found. We can then select a cut-off so that 95% of the data sets have minimum window joint probabilities above that value. We can repeat the process many times, and then take an average to get a simulated estimate of an appropriate cut-off value.

We hope that we can find some simple algorithm to calculate the cut-off for any given window width and chromosome length $L$ and number of reads N, rather than having to simulate each time. The following three graphs show the change in cut-off values as the window sizes increases from 0 to 150, for several values of N and $L$. These graphs were generated using 100 null data sets, and selecting the cut-off such that no more than 5% of data sets contained windows with probabilities below that cut-off. In this case, we kept the ratio of N/$L$ constant, simply dividing the original values used above (1/100th of the true

length of a chromosome) by 5 and by 10.

For each of these graphs, we fit the following linear model for several values of $\alpha$:

$$C_i = \beta_0 + \beta_1 w_i^\alpha + \epsilon_i$$

where $C_i$ is the cut-off value found for $w_i$, the window length. Using the lm command in R, we eye-balled the best selection for $\alpha$. The fit line $\beta_0 + \beta_1 w_i^\alpha$ is plotted with the data on each graph. Note also that the smoothness of the cutoffs increases with $L$. The plots hint that there may be some theoretical approximation we can use to find a cut-off.

### 3.2.1 Ornstein-Uhlenbeck

Note that the log probability of each window can be written as follows:

$$\log p_w = \log p(x_1) + \sum_{i=2}^{w} \log p(x_i|x_{i-1}).$$

Let us create a variable $Y$:

$$Y = 1/w \log p_w = 1/w(\log p(x_1) + \sum_{i=2}^{w} \log p(x_i|x_{i-1})).$$

If we assume that the first element in each window is distributed similarly to the later elements, then we can think of $Y$ as the average of identically distributed (but not independent) random variables. We want to model the sequence $Y$ as an AR(1) process:

$$Y_t = \rho Y_{t-1} + \epsilon$$

with residual variance $\sigma_\epsilon^2$. Then

$$\sigma^2 = \rho^2 \sigma^2 + \sigma_\epsilon^2$$

implies $\sigma_\epsilon^2 = \sigma^1(1 - \rho^2)$, where $\sigma^2$ is the marginal variance and $\rho = \text{corr}(Y_t, Y_{t-1})$.

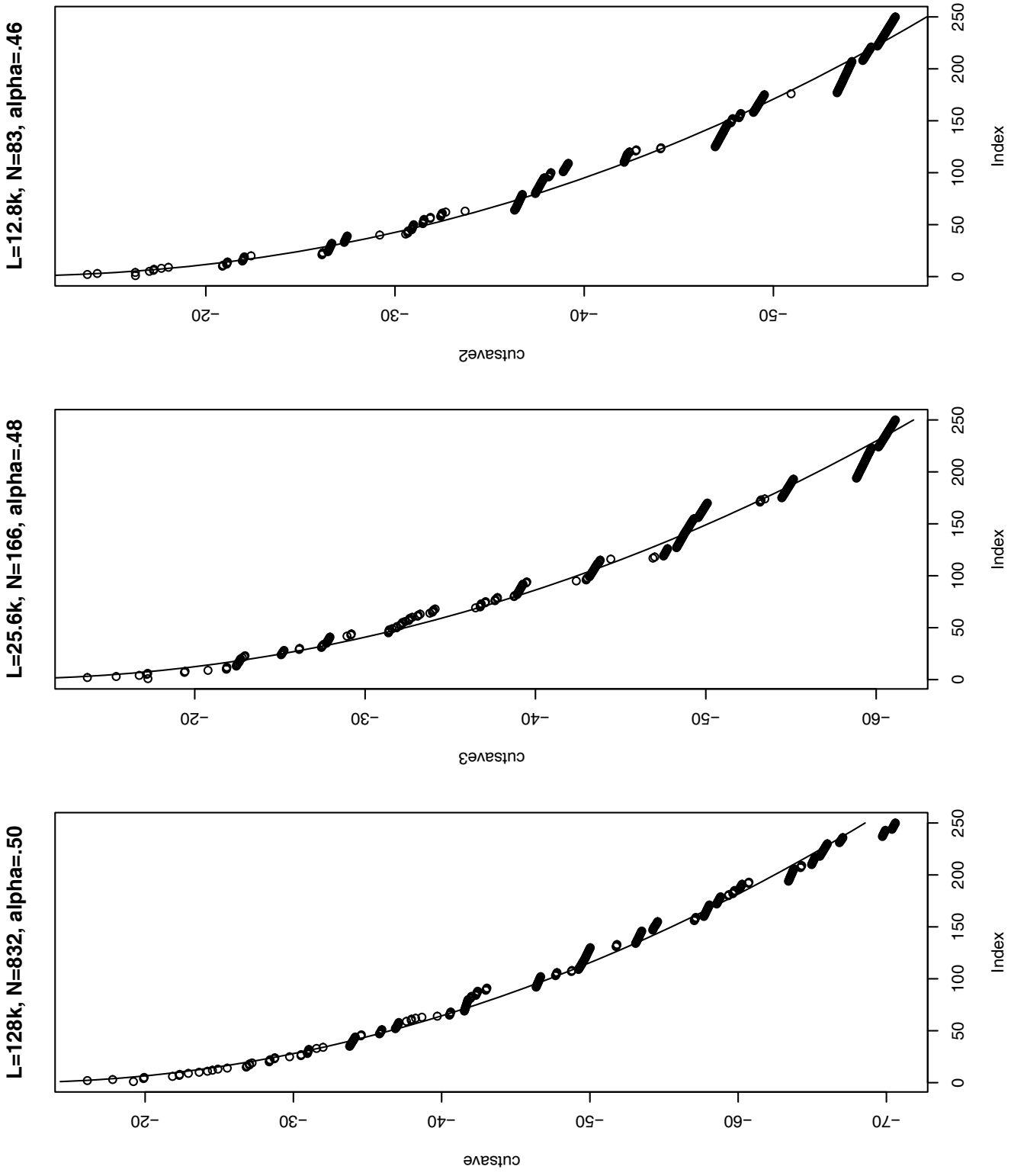


Figure 3.1: Simulated Cut-Offs for Markov Windows

It is known that the AR(1) process is the discrete time analogue of the Ornstein-Uhlenbeck process. This stochastic process satisfies the following linear stochastic differential equation:

$$dX_t = -\rho(X_t - \mu)dt + \sigma dW_t$$

where $\{W_t : t \geq 0\}$ is Brownian motion with unit variance parameter and $\mu$, $\rho$, and $\sigma$ are constants. As expected, we have the moments $E(X_t) = \mu$. $Cov(X_s, X_t) = \sigma^2/(2\rho)e^{-\rho|s-t|}$. We can thus write the related AR(1) process as:

$$x_n = \mu + \kappa(x_{n-1} - \mu) + \sqrt{1-\kappa^2}z_n$$

where $z_n$ is a random Gaussian with mean 0 and variance $\tilde{\sigma}_\epsilon^2/(2\rho)$. Here, $\kappa = e^{-\rho}$. If we define $\sigma^2$ as $\text{var}(x_n)$, we have that $\sigma^2 = 1/(2\rho)\tilde{\sigma}_\epsilon^2$.

The book "Extremes and Related Properties of Random Sequences and Processes" by Leadbetter et al. offers an estimate of the maximum of a Ornstein-Uhlenbeck process [8]. Theorem 12.2.9 of the Leadbetter book, states: **Theorem:** If $r(\tau) = 1 - C|\tau|^\alpha + o(|\tau|^\alpha)$, then for each fixed $T > 0$ such that $\sup_{\epsilon \leq t \leq T} r(t) = \delta_\epsilon < 1$ for all $\epsilon > 0$,

$$\lim_{u\to\infty} \frac{1}{u^{2/\alpha}\phi(u)/u} P\{M(T) > u\} = TC^{1/\alpha}H_\alpha,$$

where $H_\alpha > 0$ is a finite constant depending only on $\alpha$.

C is defined as $\rho = -\log(\text{Corr}(X_s, X_t))$, $\alpha = 1$ is the Ornstein-Uhlenbeck process, and $H_1 = 1$, implying that

$$P\{M(T) > u\} \sim TCu\phi(u)$$

It is possible to calculate $E(Y)$, $Var(X_t)$, and $cov(X_t, X_{t-1})$, though it takes significant computation time. Doing so with choices of $N$, the number of reads, and $L$, the genome length, at 1/10000th of the correct length gives us gives us the following graph. Each point

is for a different window size - the upper right corner is for window length 5, the lower left for window length 100. For each window, the cutoff is approximated using simulation and the Ornstein-Uhlenbeck approximation. If this was a good approximation, the points would lie on the $x = y$ line, which they clearly do not, though they are close.

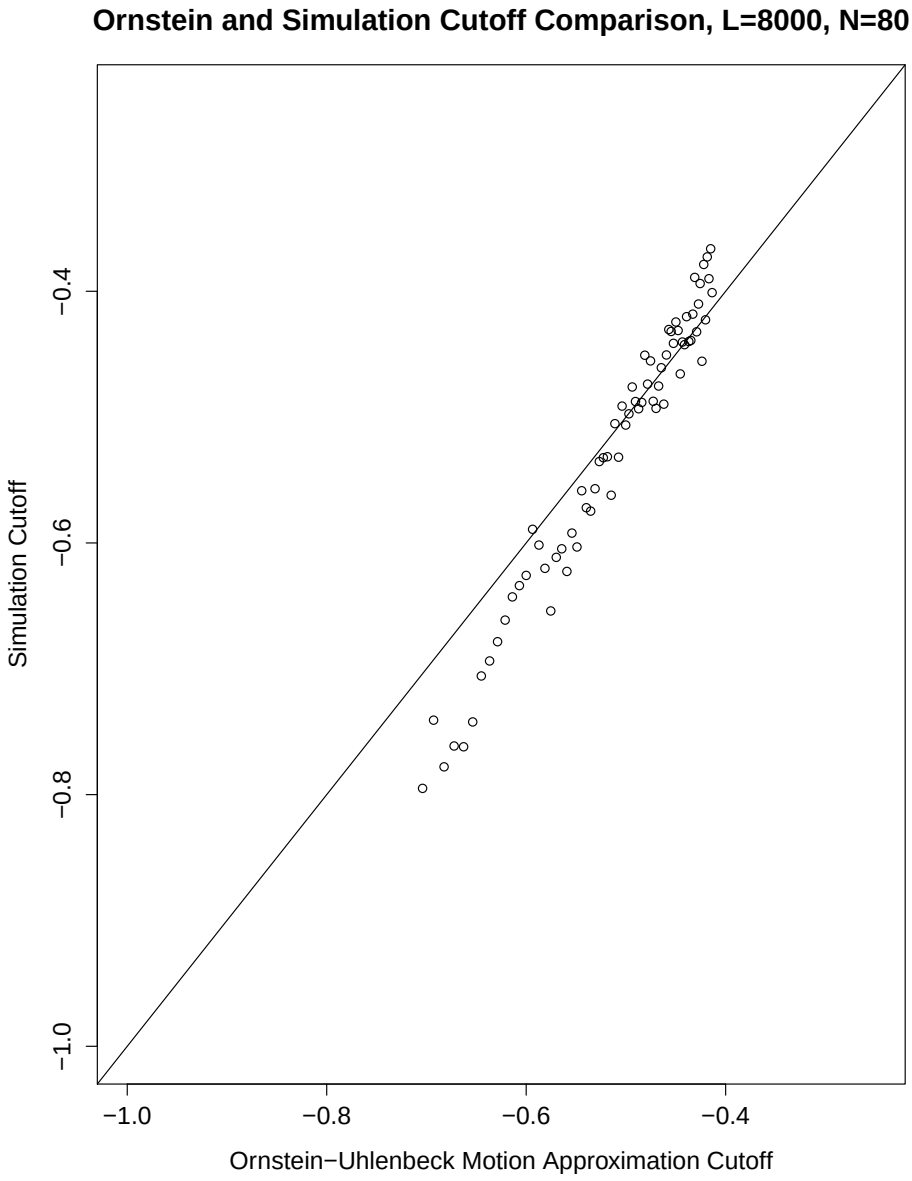


Figure 3.2: Ornstein-Uhlenbeck Approximation for N=80, L=8000

### 3.2.2 Results

The sheer size of $N$ and $L$ forced the use of the poisson approximation for the binomial. The poisson approximation of the binomial works well when N is large, $p$ is small, and $Np$ is less than 10. Since in this case, $p$ is approximately $1/L$, and for chromosome 9 $L$ is 161 million, these requirements are more than met. The first term of the Markov probability, which has neither $N$ nor $L$ is kept as a binomial, the others are approximated as a Poisson.

For $N = 80,000$, $L = 16,100,000$, one 10th the correct size of chromosome 9, this adjusted code finds the maximum of 100 reps in under 2 minutes. For the correct size, it takes significantly longer, at closer to a half hour. If we wanted several simulations to average the 95th percentile, we would have to run this overnight. A first estimate, of a single run, with a window size of 50, gives us -31.95.

Using the cut of -31.95, an $N$ of 795955, and an $L$ of 140244180, we ran this simulated cutoff on the Melnick BCL6 data for chromosome 9. The program took only an hour to run, resulting in 1,125,300 identified non-null base pairs. In comparison, ChIP-seeqer finds only 317,777. All but 65 of the ChIP-seeqer base pairs are also identified by this simulated Markov cut method. In addition, the Markov method provides a probability of each window, allowing one to rank the identified peaks by probability. The base pairs found by ChIP-seeqer had an average log probability of being null of -171.54. The base pairs found by the Markov method had an average log probability of being null of -81.19. In addition, those peak bps found by the Markov method not found by ChIP-Seqr had an average log probability of being null of -45.7, only slightly below the cutoff of -31.95. This means that the Markov method is more sensitive then ChIP-seeqer, picking up regions with lower probabilities of being non-null. ChIP-seeqer only finds those regions that are the most likely to be peak.

We can also run a check on simulated data. Using Chromosome 10 from the simulations in Chapter 5, we find that this method is not particularly effective. Chromosome 10 has a length of 135.37 million base pairs, but when we exclude all non-mappable areas, its effective length is 129.87 base pairs. The number of reads in the simulated data set are 599,511. Running the cut-off simulation with these values garners a cut-off of -28.49. With this cut-off, the Markov chain method finds 3,234 peaks, of which only 356 are true peaks in the data set. That means that there are 2,866 false peaks, and that the algorithm missed 767 true peaks. This is not a very good performance. Changing the cut-off changes the results, of course, which again opens the question of trying to identify the correct cut-off.

Rather than sticking to a thresholding method like this one, the following chapters present a model which uses a hidden Markov structure to determine peak and non-peak regions. This method helps eliminate the difficulty of selecting a specific cut-off for a data-set.

CHAPTER 4

# HIDDEN MARKOV MODELS

## 4.1 HMM Methods

HPeak, by Qin et al. (2010) [12], is a hidden Markov model method for identifying peak regions. The model is a simple HMM, fitting a zero inflated Poisson to the background and a generalized Poisson to the peak regions, and it performs well. The typical graphical representation of a hidden Markov model like HPeak is as follows:

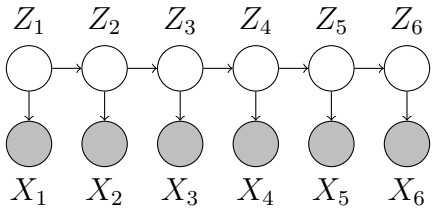


Figure 4.1: Hidden Markov Model

The hidden Markov chain (the $Z$s) in this model takes on either a 0 or 1, representing the background or peak regions respectively, and the distribution of the counts (the $X$s) depends as expected on the value of the hidden chain. In this case, the Markov dependence is only among the hidden states, thus capturing the tendency of the data to have peak regions, of many positions in a row. As such, the identification of a position as peak increases the probability of those positions around it as also being peak.

This thesis presents a model with an extra level of dependence. The data structure includes reads that are many positions long, and as such the count at any position is highly dependent on the count of the previous position. We present a model of the form below:

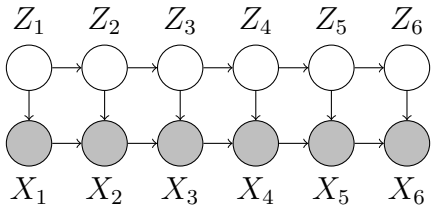


Figure 4.2: Hidden Markov Model with One-Step Dependence

The first layer of dependence among the $X$s captures the long read data structure, whereas the dependence in the hidden chain accounts for the long peak regions. In this case, while the hidden chain is still a 0-1 Markov chain, each $X_i$ depends not only on the hidden state $Z_i$ but on the previous position count, $X_{i-1}$.

To develop the emission probabilities $Pr(X_i|Z_i, X_{i-1})$, we adjust the null Markov probability model used in the Markov Chain models of chapter 3. Recall that the second binomial probability below is the probability of a new read starting at that position. In this hidden chain framework, this probability should vary dependent on whether or not the position is in a background or peak location. We can write this as follows:

$$P(x_2|x_1) \quad = \quad \sum_{j=max(0,x_1-x_2)}^{x_1} \mathrm{B}\left(j, x_1, \frac{1}{250}\right) \mathrm{B}\left(x_2 - x_1 + j, N - x_1, \pi_{z_i}\right)$$

This model leads to 4 parameters to be estimated, including the 2 parameters required for the transition probabilities across the Markov Chain.

Instead of two binomials, however, we may wish to use a Poisson. As $N$ is large and $\pi_{z_i}$, even for $z_i = 1$ is going to be quite small, the Poisson approximation fits well. The choice of a Poisson in this case has several benefits. One is the how much faster a computer can calculate a Poisson probability, thanks to the lack of a binomial coefficient. The other is the simplicity of estimating the Poisson parameters. Finally, the use of a Poisson removes the parameter $N$, the number of reads, from the equation. In the above equation, it may seem

better to use a different $N$ for peak or non-peak. By removing $N$, we do not have to account or adjust for this value. In this case, our distribution is as follows:

$$P(x_2|x_1) = \sum_{j=max(0,x_1-x_2)}^{x_1} \mathrm{B}\left(j, x_1, \frac{1}{250}\right) \mathrm{Pois}\left(x_2 - x_1 + j, \mu_{z_i}\right)$$

It is generally agreed that the Poisson distribution is a bad distribution for the count data of a ChIP-seq experiment. In chapter 1, we showed that the Poisson distribution did not fit the BCL6 data provided by the Melnick lab. Therefore, it is important to pause for a moment here and note that the Poisson distribution is not being used to estimate the counts. Instead, the Poisson distribution is being used to estimate the peak and non-peak distribution of read starting positions - that is, the Poisson distribution calculates the probability of reads starting at any given position. In this paradigm, a read is only counted if it begins at the specific position in question. Unlike the full counts, the read start counts are in fact independent, and therefore, the Poisson distribution is not unreasonable for an initial distribution.

From HPeak, we know that the EM Baum-Welch algorithm, usually used for parameter estimation in a hidden Markov model, is not a possibility for data sets of this size. The format of this joint distribution also prevents finding a maximum likelihood estimate for $\mu_{z_i}$. Instead, we follow HPeak and run an iterative Viterbi-Moments estimate. The Viterbi algorithm can be coded efficiently, and unlike the Baum-Welch, requires only a single pass through the data. The question, then, is how to estimate the four parameters.

## 4.2 Model Fitting and Parameter Estimation

HPeak fits its model by alternating between the Viterbi Algorithm, which, when given the model parameters, identifies the hidden chain with a best fit of 0s and 1s, and method of moments estimates, which can calculate the model parameters when given the hidden chain.

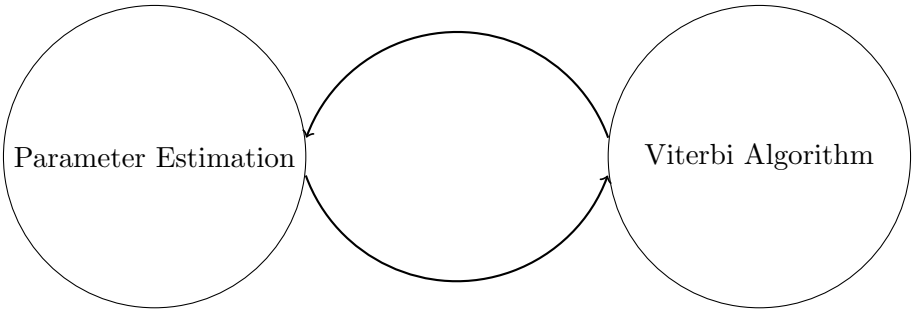


Figure 4.3: Model Estimation

The Viterbi Algorithm, developed by Andrew Viterbi [**?**], finds the sequence of 0s and 1s that maximizes the total likelihood of the model. It is a recursive algorithm that finds a weight $v(z_i)$ for each position as follows: let $v(z_i)$ be the weight of the most likely path for $(x_1, \ldots, x_i)$ that ends in state $z_i$. Then,

$$v(z_i) = p(x_i|z_i) \max_{z_{i-1}} v(z_{i-1}) \lambda_{z_{i-1},z_i}$$

where $\lambda_{z_{i-1},z_i}$ is the transition probability of the Markov chain. For each possible value of $z_i$ (0 or 1), $v(z_i)$ is calculated, and the value of $z_{i-1}$ that satisfies the maximum in the equation above is recorded. This record simply states the most likely value of $z_{i-1}$ for each possible value of $z_i$, and, once the recursion is run, a simple trace-back through this record generates the most probable list of zeros and ones. When $x_i$ depends on $z_i$ and $x_{i-1}$, as it does in our model, the Viterbi algorithm still holds with only a slight adjustment.

To calculate the Poisson parameters $\mu_{z_i}$, we need to develop a method of moments estimate. Recall that $\mu_{z_i}$ represents the average number of new reads beginning at any specific

position when the hidden Markov chain is equal to $z_i$. When the Viterbi algorithm is run, we have a set of positions with the hidden Markov Chain equivalent to a specific value. We can then calculate

$$\mu_1 = \frac{\text{\# reads starting in non-null region}}{\text{length of non-null region}} \qquad \mu_0 = \frac{\text{\# reads starting in null region}}{\text{length of null region}}$$

How do we determine the number of reads starting in each non-null region? The data is arranged such that every read is exactly the same determined length, so one pass through the data can calculate the exact starting locations of every read. Therefore, this number can be found exactly from the count data.

The question that remains is how to calculate estimates for $\lambda_0$ and $\lambda_1$, the transition probabilities of the hidden Markov chain, where $\lambda_i$ is the probability of transitioning to state $i$ given that the chain is in state $i$. One obvious estimate is simply:

$$1 - \lambda_1 = \frac{\text{\# of transitions from 1 to 0}}{\text{\# of positions equal to 1}}$$

The estimate for $\lambda_0$ is equivalent.

HPeak, by Qin et al. [12], uses a different estimate. They take advantage of the fact that the median length of any peak region should follow the geometric distribution, so that they have:

$$\lambda_1^L = 1/2,$$

where $L$ is the median length of the peaks, easily estimated from the data. From this, an estimate of $\lambda_1$ can be found. As for $\lambda_0$, they set $1 - \lambda_0$, the probability of transition from non-null to null, to be the proportion of the genome covered by the peaks. They also use this probability as the initial probability of being non-null.

Running the Viterbi algorithm the first time requires initial parameter estimates. This can be done with some guesswork based on the overall averages. Or, one can use a cut-off

method to select peaks and background and estimate parameters from this. Below, the cut-off is set somewhat arbitrarily to be the mean count plus twice the standard deviation.

## 4.3 Preliminary Simulations

To check the efficacy of the model described above, we developed a method for simulating data. Peak areas were selected, and reads were assigned to be either peak or background with some probability. If background, the read had an equal probability of starting at any background position. If peak, the read had an equal probability of starting at any peak position or any position less than the read length prior to a peak position.

For the simulations below, we used chromosomes of length 80,000, with varying numbers of peaks, read lengths, and number of reads. The peaks were relatively strong, with the reads having a 30% chance of being in the peak regions versus the non-peak regions. We then ran the following algorithm:

**Step 0:** Find and record all starting positions of reads.

**Step 1:** Select cut-off, assign initial peak and non-peak values.

**Step 2:** Estimate parameters $\pi_1, \pi_0, \lambda_0, \lambda_1$ using Markov chain peak/non-peak values.

**Step 3:** Run Viterbi algorithm using parameter values to redetermine peak/non-peak chain.

Iterate steps 2 and 3 until convergence.

Note that the algorithm above is interested in read starting positions, not in the highest count positions, meaning that the algorithm picks up the start of each peak region, and not the middle or end. To solve this directionality problem, we perform the following steps:

**Step 1:** Run algorithm on sequence of counts, record peak region results.

**Step 2:** Reverse full sequence of counts.

**Step 3:** Run algorithm on reverse sequence.

**Step 4:** Take the union of forward and reverse runs as peak regions.

With this algorithm, the result is a symmetric peak region around the peak position. Finding the center of the region finds the actual binding site. To account for this, we add one more step:

**Step 5:** Extend each forward and backward peak region directionally by the peak length, so as to fill in the center of the full joint region.

We call the method described above HiDe-Peak, for HIdden Markov model with DEpendence for Peak finding.

## 4.4 Simulation Results

The first simulation run had 5 peaks, read length 250, 80 reads, and chromosome length 80,000. Each read had a 30% probability of being a peak read. Below is an image of the values, with the circles representing those areas initially declared as peak using the mean plus 2 standard deviation cut-off. Running this simulation resulted in the entire region being called peak. To adjust for this, we set the cut-off much higher - to the mean plus six standard deviations. In this case, the algorithm found 8 regions, four upstream of the binding sites and four downstream of the binding sites. Two of the protein binding sites were quite close

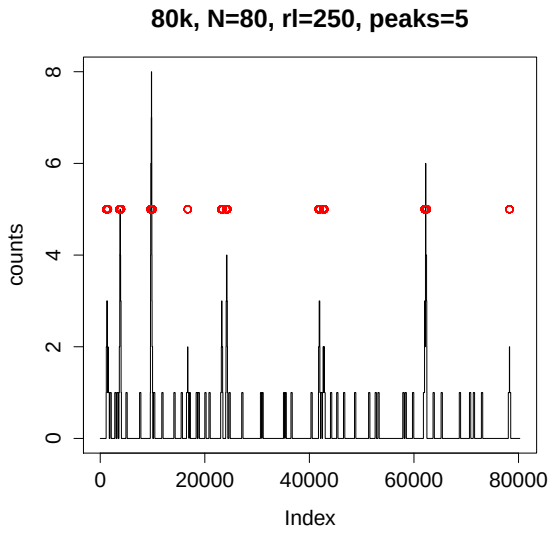


Figure 4.4: First Simulation

| | | | | |
|---|---|---|---|---|
| binding site(s) | 3814 | 9791 | 23278 24183 | 62229 |
| upstream region | 3579-3800 | 9575-9790 | 23994-24394 | 62023-62225 |
| downstream region | 3829-4050 | 9825-10040 | 24244-24394 | 62273-62475 |
| midpoint | 3814.5 | 9807.5 | 24194 | 62249 |

Table 4.1: Simulation 1 Results

(within 1000), and so the algorithm only found one large region containing both. In general, the method did well.

This case is not actually as accurate a case of the real data as we might wish. Note that these peaks are relatively weak, with a highest count of 8. In addition, the true data would rarely if ever have so many peaks in a region 80k long. We can run this with a single peak, though by making the peaks less common, the single peak is much stronger. Below is an example of this case. This is much easier for the algorithm, as the single peak is very visible. Once again, it identifies two regions, one downstream and one upstream of the peak. The midpoint of these two peaks is 15741.5, only 1.5 past the true binding position of 15740.

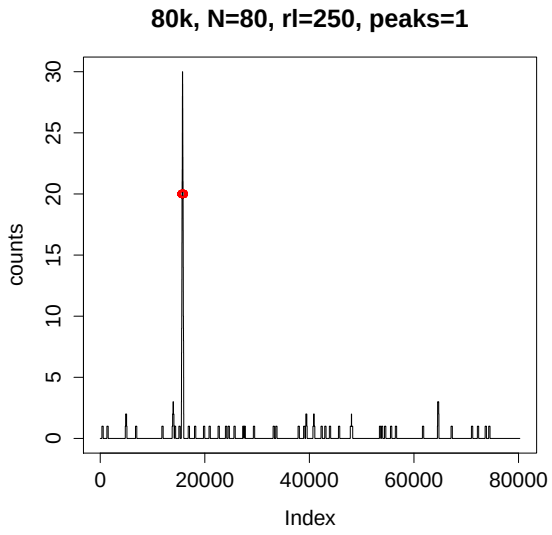


Figure 4.5: Second Simulation

The second simulation that might be of interest is to consider 'binning' the data, as Qin et al. do in HPeak. [12] Binning has the dual advantages of reducing the dependence due to read length, and reducing the total amount of data so as to decrease computational time. If we create bins of ten consecutive genome positions, then our effective read length is 25. Our chromosome length of 80k is now 800k, so the number of reads should be 800, not 80. With these changes (and with five peaks) we get data of the following form. Once again, the circles mark areas initially determined as 'peak.'

In this case, with the shorter effective read length, five full regions were found, bracketing the true binding position. In fact, the midpoints of these regions were no more than half a position away from the true binding site. Each of these regions were almost exactly 50 positions in length, thereby finding the 25 positions before and 25 positions after that make up the peak region.

It is true that in this case the peaks were very strong. We can make them weaker

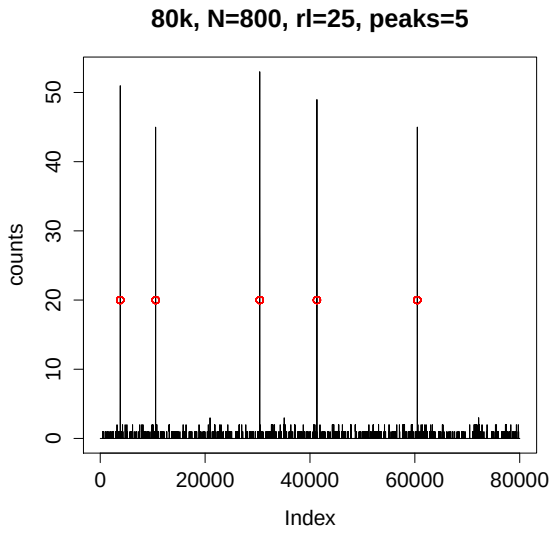


Figure 4.6: Third simulation

| | | | | | |
|---|---|---|---|---|---|
| binding site(s) | 3820 | 10562 | 30419 | 41323 | 60472 |
| found region | 3796-3845 | 10537-10587 | 30394-30444 | 41298-41348 | 60447-60497 |
| midpoint | 3820.5 | 10562 | 30419 | 641323 | 60472 |

Table 4.2: Simulation 3 Results

by reducing the probability of a read being declared 'peak' from 30% to something much weaker, such as 5%. Because the peaks are much weaker, we set the cut-off to be mean plus 5 standard deviations, not 2. The plot, along with the initial regions selected using this higher cut-off, is below.

Note that the initial setting picks up several false peaks, but the algorithm succeeds in weeding those out of the final peaks. With these weaker peaks, the algorithm finds more narrow regions with pairs of regions upstream and downstream of the binding position. In this case we can see that the algorithm found one false peak and missed a true peak. Those peaks it found correctly, though, it did very well with, finding a midpoint only 1 or 2 away from the true binding location in each case.

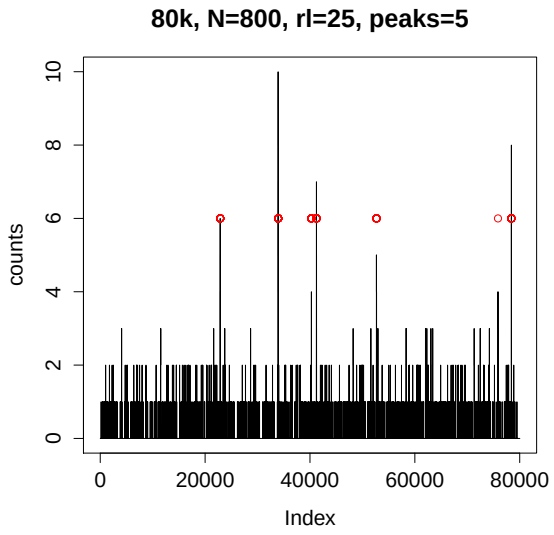


Figure 4.7: Fourth Simulation

| | | | | | |
|---|---|---|---|---|---|
| binding site(s) | 22879 | 33927 | 41238 | 52694 | 78416 |
| upstream region | 22854-22876 | 33902-33925 | 40257-40263 | 41216-4132 | 78392-78411 |
| downstream region | 22879-22901 | 33927-33950 | 40282-40288 | 41241-41257 | 78417-78436 |
| midpoint | 22877.5 | 33926 | 40272.5 | 41236.5 | 78414 |

Table 4.3: Simulation 4 Results

Running this again with another data set but with the same parameters, the algorithm correctly identified all five peak regions, suggesting that even with weak peaks, the algorithm can perform optimally. Once again, the original peaks included one false peak, but the algorithm correctly identified it as null. These small simulations assume a background model equivalent to the model we are using, so it is not surprising that this method is so successful. More complex simulations and described in chapter 5.

## 4.5 Comparisons to HPeak

HPeak, by Qin *et al.* (2010) [12], is a popular existing peak finding method that uses a standard two-state HMM. Their model for transition probabilities is the same as used in our HMM method, but they assume different emission probabilities. The authors used the Generalized Poisson (GP) and the Zero Inflated Poisson (ZIP) distributions to model the counts within each bin. The GP distribution has two parameters, and is:

$$P(Y = y|\lambda, \phi) = \left(\frac{\lambda}{1+\phi\lambda}\right)^y \frac{(1+\phi y)^{y-1}}{y!} \exp\left\{\frac{-\lambda(1+\phi y)}{(1+\phi\lambda)}\right\}$$

The Zero Inflated Poisson is used for the background distribution, due to the many bins with 0 counts. It also has two parameters:

$$f(x|\pi, \mu) = \begin{cases} (1-\pi) + \pi e^{-\mu} & \text{if } x = 0 \\ \frac{\pi e^{-\mu}\mu^x}{x!} & \text{if } x > 0 \end{cases}$$

where $\pi$ is the proportion of zeros in the mixture distribution. They use the method of moments to estimate the parameters of these distributions, then iterate with a Viterbi algorithm to find the peak regions.

The HPeak software is available for download, but as it is written with a combination of Perl scripts and C++ code and is designed to handle data sets in a different format than the simulated data and the data provided by the Melnik lab, the version of HPeak used below is coded in R and based on the description in their paper.

Each simulation assumes a genome of length 80,000, with 800 reads of read length 25. This is approximately equivalent to assuming a genome of ten times that length, summing across bins of length 10, with reads of length 250. The probability of a read being selected to fall in a non-null region was .25, making strong peaks. Each simulation below contains an image of a single simulated data set, where it is clear that the peaks are strong.

Finally, to account for the fact that typically not all peaks are the same strength, some data sets allowed the peaks to vary in strength. To do, we simply simulated weights from a uniform distribution, which we then scaled and applied to each peak. Once a peak was determined to be non-null (using the .25 probability), it was then assigned to each of the peaks according to the simulated weights. This allowed the peaks to vary in height by a random process.

Finally, the convergence criterion was .0001, so that between iterations, the parameters could not change by that amount. The initial cut-off used for both methods was 3 standard deviations above the mean.

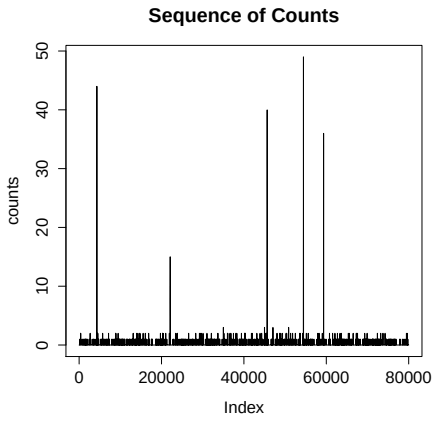


Figure 4.8: 5 Peaks, Random Sizes

| 5 Peaks, Random Sizes | | | |
|---|---|---|---|
| Method | Mean # of Peaks | Mean # Correct Peaks | Mean Peak Length |
| HiDe-Peak | 4.88 | 4.80 | 50.31883 |
| HPeak | 318.5 | 4.91 | 30.36452 |

Table 4.4: HPeak Comparison 1

It is nice to see that HiDe-Peak performs very well in these simulations, even with the peaks being of varying heights. However, note that the number of peaks for HPeak is very

| 10 Peaks, Random Sizes | | | |
|---|---|---|---|
| Method | Mean # of Peaks | Mean # Correct Peaks | Mean Peak Length |
| HiDe-Peak | 9.35 | 9.14 | 47.76782 |
| HPeak | 410.03 | 9.54 | 30.06208 |

Table 4.5: HPeak Comparison 2

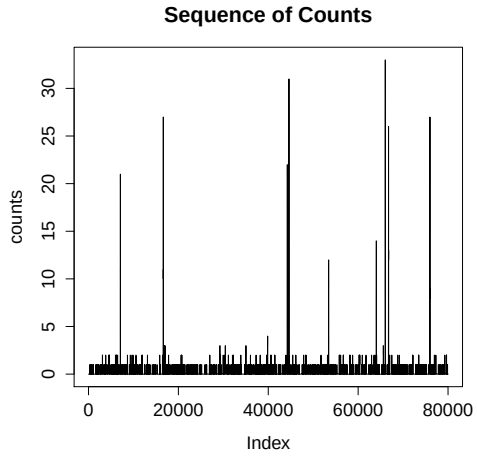


Figure 4.9: 10 Peaks, Random Sizes

high. Looking at the individual results for the 100 simulations, we find that in 90 of the simulations, the number of peaks was well over 400, in which the algorithm declares all non-zero areas to be peaks. In the remaining 10, the number of peaks was between 65 and 100. The cases with above 400 peaks have trouble estimating the zero-inflated Poisson distribution, finally converging to $\hat{\pi} = 1$ and $\hat{\mu} = 0$. This may be due to the fact that the background distribution is not actually a zero-inflated poisson, due to the way that the data is simulated. When we assume binding sites of equal strength, HPeak performs better, with only about 50 of the simulations ending up in the extreme case of $\hat{\pi} = 1$. The other fifty simulations result in around 80-100 peaks.

To determine the exact problem with HPeak, we examined the method of moments

| 10 Peaks, Random Sizes | | | |
|---|---|---|---|
| Method | Mean # of Peaks | Mean # Correct Peaks | Mean Peak Length |
| HiDe-Peak | 10.42 | 10.00 | 48.39309 |
| HPeak | 309.51 | 10.00 | 26.53233 |

Table 4.6: HPeak Comparison 3

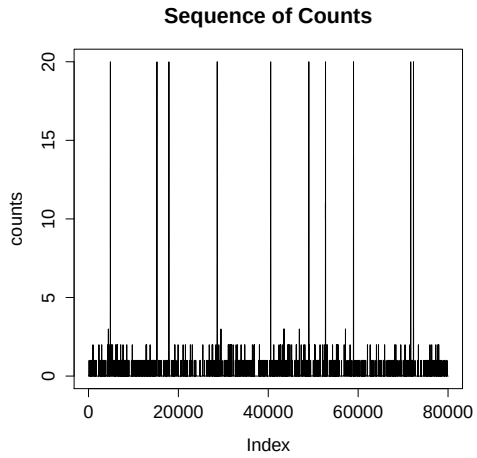


Figure 4.10: 10 Peaks, Equal Sizes

estimates for the zero-inflated Poisson more carefully. They are as follows:

$$\hat{\mu} = \frac{S^2}{\bar{X}} - 1 + \bar{X}, \qquad \hat{\pi} = \frac{\bar{X}}{\hat{\mu}}$$

Note that if the sample variance is not more than the sample mean, $\hat{\mu}$ will be less than $\bar{X}$. This means that $\hat{\pi}$, meant to represent the extra proportion of zeros in the data, will be greater than 1. From there, the algorithm cannot converge correctly. To account for this, we added in a small amount of code to use the Poisson, and not the ZIP, in the case where $\bar{X} > S^2$. However, doing so did not solve the problem. The code still declares all non-zero areas to be peaks in most cases.

Examining the code for HPeak 2.1, available on author Steve Qin's website reveals that the code differs from the paper in several ways. The transition parameters are being esti-

mated as stated in the paper, with the peak-to-background depending on the median peak length, and the background-to-peak depending on the percent of peak base pairs in the current chain. The estimate of $\lambda_0$ in the code appears a bit odd, however, the result of a possbile unfixed debugging attempt, in that it is $1 - TotalPeakArea/(3100 * .09)$. We want $\lambda_0$ to be the proportion of the sequence that is null, so it should be $1 - TotalPeakArea/L$, where $L$ is the length of the genome. Why 3100*0.9 then? Assuming that is an old error, the rest of the estimation appears the same as stated in the paper.

The emission parameters are being estimated quite differently than the paper. Instead of the Zero-Inflated-Poisson (ZIP) and Generalized Poisson, HPeak 2.1 instead uses two Generalized Poisson (GP) distributions, but the distribution for the peak regions is a zero-truncated GP. To find the parameters for the background GP, HPeak 2.1 simply uses the two method of moments estimates. But for the peak distribution, the algorithm uses a Gibbs sampler to sample from the distribution, and then selects the parameters such that the likelihood is maximized.

When running the Viterbi algorithm, the method calculates $p_0$ to be the generalized Poisson using the method of moments estimators, and $p_1$ to be the zero-truncated-Poisson using the Gibbs MLE estimators. For stability purposes, he then adjusts $p_0$ to be $p_0/(p_0+p_1)$ and $p_1 = p_1/(p_0 + p1)$. Setting them this way does not changed the algorithm, and allows the algorithm to set $p_1$ to 1 when the counts are very high (above some chosen limit) and $p_0$ to be declared 0 when the counts are zero. This spares long computations, and as much of the sequence is zero, it speeds up the algorithm.

Finally, in the traceback part of the Viterbi algorithm, Qin appears to not do the traditional traceback. Instead, HPeak 2.1 calculates a ratio based on the two probabilities found by the Viterbi algorithm: $\Pr(y_i|Z_i,\ \text{path})$, where the path is the most probable path through

the hidden Markov Chain such that it ends in $Z_i$, where $Z_i$ is the value of the hidden Markov chain at position $i$. Qin calculates the ratio of the probability when $Z_i = 0$, adjusted for the transition probability from 0 to the next state, to the total when $Z_i = 1$ or 0. He then selects the path to be 0 at that position with probability equal to that ratio, and 1 otherwise. This prevents the computational complexity of a full traceback.

Currently, the code only does this method for a single iteration, not running until convergence as was stated in the paper. Dr. Qin claims that this is a de-bugging error, which he will fix. However, the code does not test for convergence at all - instead, it runs a number of iterations that is pre-set by the user.

This code is different significantly from the method described in the paper, likely because this is a working program under constant revision. Unfortunately, when this code is transferred into R to run on the test data sets, it still runs into the same convergence issues as before. Like the method described in the paper, this method also likes to converge to the case where all non-zero locations are declared to be peak. In the comparisons below, we simply run the algorithm three times, as would be done in the real case, and run the algorithm on a variety of data sets.

## 4.6 Simulations with noisy background data

One issue with the simulations above was the lack of 'noise' in the background of the peaks. It is known that ChIP-seq data has a good deal of noise, and that reads do not necessarily align themselves evenly across the background. To account for this, we added weights in the sampling method for the background data. The random data creation function now randomly selects $N$ null starting positions, then places weights on each of those positions

from a beta distribution, with varying parameters $\alpha$ and $\beta$. The null reads are then selected from this subset, using the weights, so that some background positions will be more likely ("stickier") then others.

Below is the first of several simulations comparing HPeak (based on version 2.1) and HiDe-Peak. The HPeak code was not run to convergence, simply for 3 iterations. There were 5 peaks, with the probability of a read selecting a peak region quite low at 0.05. As usual, the chromosome is length 80,000, there are 800 reads, and the read length was 25. The noise in the background was set with $\alpha = 2$ and $\beta = 5$. You can see that the reads are still quite strong, but the algorithm does much worse, identifying many more peak regions than just the main peaks. However, it does out-perform HPeak, which finds far more peaks.

| 5 Peaks, Random Sizes, Noisy Background (2,5) | | | |
|---|---|---|---|
| Method | Mean # of Peaks | Mean # Correct Peaks | Mean Peak Length |
| HiDe-Peak | 177.5 | 4.5 | 27.3 |
| HPeak | 406.1 | 4.7 | 26.3 |
| 5 Peaks, Random Sizes, Noisy Background (2,2) | | | |
| Method | Mean # of Peaks | Mean # Correct Peaks | Mean Peak Length |
| HiDe-Peak | 181.1 | 4.0 | 27.3 |
| HPeak | 426.8 | 4.4 | 25.9 |

Table 4.7: Noisy Simulations

The difficulty of separating out the strongest reads from the weakest suggests some possible future work, in which the hidden Markov chain has more than two states. For example, the hidden Markov chain can have strongly peak, weakly peak, and null. Another solution, explored in Chapter 5, is to switch to an over-dispersed distribution like the Negative Binomial to account for the variety of peak strengths.

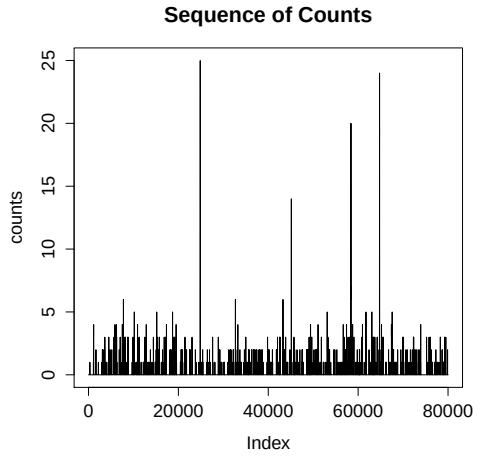


Figure 4.11: 10 Peaks, Equal Sizes

## 4.7 Computational Considerations - Python/Viterbi

The natural extension after dealing with these simulations is to move the code into Python, which can handle larger data sets. There are several computational considerations, most having to do with memory issues. In the original R code, a vector of length $l$ was created to keep track of each position and whether or not it could be considered peak or non-peak. Now that $l$ is up to 247 million, the creation of such a vector is no longer viable. Now the code records not each position's peak status, but merely the start and end of each peak region.

A second, more major problem, was the need to read the Viterbi algorithm table backwards for the trace-back at the end of the algorithm. While there exist many ways to read a file in backwards in Python, these were all cumbersome and quite slow. It is not something Python does naturally. The trace-back table consists of two columns of 0s and 1s, with each column being the $\max z_{i-1}$ for each possible value of $z_i$. When $\max z_{i-1}$ is the same for both $z_i = 0$ and $z_i = 1$, then the method can be trace-backed from there, and the peak positions can be recorded. As this occurs frequently, the amount of the trace-back table that needs to

be recorded or read at any time is relatively small. This has the added benefit of eliminating the need to store the 247 million by 4 Viterbi table in a file on the hard drive.

With these considerations, the algorithm can handle very large data sets, though the algorithm is still slow. If we were to bin the data, as HPeak does, the speed would improve considerably. Naturally, it would also improve the speed to run this code on a computer that was not a personal Dell laptop. However, the fact that it can run on a laptop (which HPeak cannot), is something to be said in its favor.

## 4.8 Results on real data

This algorithm was run on the BCL6 data from the Melnick lab for chromosome 9, the same chromosome examined in chapter 3. The algorithm took 30-some hours and 16 iterations to converge, starting with an initial cutoff of 4 standard deviations above the mean. This looked too low, as the average count for the peak regions rose from 1.34 to 4.49 by the time it converged. Running it again with a cutoff of 8 standard deviations above the mean caused the algorithm to converge faster, but to the same value, which is a quick spot check that the algorithm is at least a little robust to the starting values. Select parameter estimates over the 16 iterations of the first run are printed below. As you can see, $\lambda_1$ dropped to .5, implying that more than half of the peaks were of width one. The algorithm finds areas of concentrated starts of peaks, so these should be narrower than the full peak length, but a size of 1 is still very low.

| Iter | $\mu_0$ | $\mu_1$ | $\lambda_0$ | $\lambda_1$ |
|---|---|---|---|---|
| 1 | 0.05924 | 1.34560 | 0.99591 | 0.98636 |
| 2 | 0.06117 | 2.30079 | 0.99851 | 0.97753 |
| 3 | 0.06169 | 2.80085 | 0.99897 | 0.94387 |
| 4 | 0.06194 | 3.09076 | 0.99915 | 0.87055 |
| 5 | 0.06209 | 3.30607 | 0.99926 | 0.70710 |
| 10 | 0.06271 | 4.45629 | 0.99959 | 0.5 |
| 15 | 0.06273 | 4.49414 | 0.99960 | 0.5 |
| 16 | 0.06273 | 4.49414 | 0.99960 | 0.5 |

We compared the results to ChIP-seeqer. ChIP-seeqer identified 317,377 peak positions. We were unable to run this in both directions, as desired, due to time and computability constraints, but we did extend the HMM method peaks all by the length of the reads, since the peaks are centered around the starting positions of each read. Once this was done, this method found 109,722 peak positions, of which 68,987 (or 68%) were also identified by ChIP-seeqer.

We can compare this to more than just ChIP-seeqer by using the STAT1 data set used in a variety of papers (including HPeak). The STAT1 data set, used first by Robertson and Mortazavi, is one of the first published ChIP-seq data sets, and is available freely online. The reads are, on average, 150 in length, and the data set is in the common ELAND format.

The parameters were as follows across the last three iterations. In total, it took only 5 iterations to converge:

Note that $\lambda_1$, the probability of staying in a peak region if already in a peak region, dropped to 0.5 quickly. This is due to the fact that the median peak length was 1. In the results above, all the peaks of size 1 were removed (that is, they have length 0, both starting

| Iter | $\mu_0$ | $\mu_1$ | $\lambda_0$ | $\lambda_1$ |
|---|---|---|---|---|
| 3 | 0.0020 | 0.3105 | 0.9997 | 0.5 |
| 4 | 0.0020 | 0.3214 | 0.9997 | 0.5 |
| 5 | 0.0020 | 0.3220 | 0.9997 | 0.5 |

Table 4.8: Convergence Iterations for Stat1 Data set

and ending at the same position). Here are the results, comparing only to HPeak:

| Method | Number of Peaks | Covered space (kb) | Avg Peak width (bp) |
|---|---|---|---|
| HPeak-b | 43,443 | 15,354 | 353 |
| HIDE-Peak each Chr, no length 0 | 82,929 | 14,841 | 180 |
| HIDE-Peak shared Params | 611,927 | 98,720 | 163 |
| HIDE-Peak shared Params, no length 0 | 54,741 | 10,258 | 188 |
| HIDE-Peak shared Params, Matched Peaks | 24,334 | 16,666 | 688 |

Table 4.9: Stat1 Comparison with HPeak

The first column estimates the parameters separately for each chromosome. Although this is faster, as the chromosomes can be fit in parallel, it leads to over-fitting. The experiment is performed on the whole cell, and not on a chromosome-by-chromosome basis. It is clear here that the results are quite different when calculating each chromosome separately, and so the added computational complexity of sharing parameters is worth he time.

The first shared parameters row includes all those peaks that are length 0, and this covers a significant portion of the genome. Excluding those peaks, we get something much more in line with HPeak. The final row only looks at "Matched Peaks," that is, all those peaks on the forward strand that have a peak on the reverse strand at least 400 base pairs downstream. Because we extend the length of the peaks up to 400 to find their matched pair, we do get on average much longer peaks. But the number of peaks drops, and the results still look to be in line with HPeak. Using matched peaks follows some of the ideas in Valouev et. al [14] as well as other methods, though later simulations show that matched peaks may be too restrictive.

Why matched peaks? This algorithm is extremely sensitive to a set of reads starting together, and as a result identifies the beginning and end of peak regions much more readily then the middle, where the protein binding site is actually located. As such, a true protein binding region should have a peak on the forward strand, and then a matching peak downstream on the reverse strand, approximately one read length apart. Peaks only on one strand are are more likely to due to experimental noise - things that should be accounted for by a control data set. So only using pairs of matched peaks helps eliminate false peaks, especially when working in the absence of control data.

It's important to note here that the initial cut-off used in the analysis above is actually quite high. There is a position on chromosome 9F with several thousand reads starting at that position. As a result, the variance for chromosome 9F (where the F notes that we are on the forward strand, not the reverse), is 1542. Chromosome 3F has variance 17.02, 23F has 3.41, and 2R has 1.76, but all of the others have a variance of less than 0.5. Including 9F in the estimation of the genome-wide variance makes the overall variance 38, with the standard deviation at 6.17. However, since this position on chr 9F appears extremely anomalous, it makes sense to exclude it. Doing so gets us a variance of .9609, much more in line with the results. Recalculating the cut-off using this new standard deviation gets us a cut-off of 2.935, significantly lower then the 18.56 used in the analysis above.

While this took much longer to converge, the number of 0-length peaks is much lower. In fact, the median peak length is 16 in this case, not 1 as in the case with the higher peak, indicating that the initial peak selection has a large effect on the results. Here is how the peak counts come out with this initial cut-off value of 2.9, along with comparisons to a wider variety of methods. The results of other methods listed here come from the analysis done by HPeak [12].

| Method | Number of Peaks | Covered space (kb) | Avg Peak width (bp) |
|---|---|---|---|
| MACS | 22,402 | 16,940 | 756 |
| FindPeaks | 41,127 | 46,781 | 1,137 |
| HPeak-b | 43,443 | 15,354 | 353 |
| SISSRs | 9,561 | 455 | 10,012 |
| CisGenome | 38,878 | 10,012 | 258 |
| Mosaics | 18,833 | 10763 | 546 |
| HPeak-b | 43,443 | 15,354 | 353 |
| HiDe-Peak Each Chromosome | | | |
| No length 0 | 82,929 | 14,841 | 180 |
| HiDe-Peak Shared Parameters, cut-off 17 | | | |
| All Peaks | 611,927 | 98,720 | 163 |
| No length 0 | 54,741 | 10,258 | 188 |
| Matched Peaks | 24,334 | 16,666 | 688 |
| HiDe-Peak Shared Parameters, With cut-off 2.9 | | | |
| All Peaks | 169,763 | 35,838 | 213 |
| No length 0 | 137,998 | 30,942 | 225 |
| Matched Peaks | 13,514 | 11,547 | 847 |

Table 4.10: Stat1 Results Table

These results vary significantly across the many methods, and it is hard to tell the accuracy of these results when only looking at real data, since the true answer is not known. However, this method does not appear to be very much in line with the competition. One concern may be the use of the Poisson distribution.

## 4.9 Generalized Poisson

Using the peak and non-peak regions of the Stat1 defined by HiDe-Peak as described in the previous section, we can fit a variety of distributions and test for goodness-of-fit.

The first thing to check is if the distribution of reads is Poisson in peak regions. We only count read starts, not read counts, in this model, the assumption of independence ought to

| # Starts | Observed | Expected |
|---|---|---|
| 0 | 9666273 | 9453662 |
| 1 | 834525 | 1169071 |
| 2 | 154430 | 72286 |
| 3 | 26560 | 2980 |
| $\geq$4 | 16287 | 94 |

Table 4.11: Poisson Fit for Peaks

| | | | |
|---|---|---|---|
| 0 | 9666273 | 10-19 | 1126 |
| 1 | 834525 | 20-29 | 159 |
| 2 | 154430 | 30-39 | 57 |
| 3 | 26560 | 40-49 | 21 |
| 4 | 7548 | 50-59 | 16 |
| 5 | 3361 | 60-69 | 10 |
| 6 | 1777 | 70-79 | 6 |
| 7 | 1059 | 90-99 | 5 |
| 8 | 652 | $\geq$ 100 | 18 |
| 9 | 486 | | |

Table 4.12: Distribution of Peak Counts

hold, which makes the Poisson a not unreasonable choice. Using $\lambda = 0.1236633$, which is the estimated $\lambda$ from the data, we create a table with the observed and expected counts, for a Pearson's chi-squared test:

The chi-squared value for this table is 3169948, so we clearly reject the hypothesis that this is Poisson. While it is true that you will reject any test of a distribution with enough data, this is a particularly bad fit. To give you an idea, we had to cap the comparison at 4 because we didn't have any 'expected' values greater than 4. But we had plenty of observed values greater than 4:

The distribution appears to be much more heavy-tailed, like an over-dispersed Poisson. As such, we try a negative binomial. We find with p=0.5066833 and r=0.1204010, both found using the method of moments. As we can see, the negative binomial is heavier tailed, but not heavy-tailed enough. We find a chi-squared value of 175449.4, which is better, but

| # Starts | Observed | Expected | # Starts | Observed | Expected |
|---|---|---|---|---|---|
| 0 | 9666273 | 9825585 | 8 | 652 | 869 |
| 1 | 834525 | 599412 | 9 | 486 | 397 |
| 2 | 154430 | 170140 | 10 | 325 | 184 |
| 3 | 26560 | 60931 | 11 | 192 | 86 |
| 4 | 7548 | 24084 | 12 | 145 | 40 |
| 5 | 3361 | 10056 | 13 | 122 | 19 |
| 6 | 1777 | 4348 | 14 | 102 | 9 |
| 7 | 1059 | 1926 | $\geq 15$ | 536 | 7 |

Table 4.13: Negative Binomial Fit for Peaks

| # Starts | Observed | Expected | # Starts | Observed | Expected |
|---|---|---|---|---|---|
| 0 | 9666273 | 9808101 | 9 | 486 | 552 |
| 1 | 834525 | 632598 | 10 | 325 | 284 |
| 2 | 154430 | 160214 | 11 | 192 | 149 |
| 3 | 26560 | 55808 | 12 | 145 | 79 |
| 4 | 7548 | 22496 | 13 | 122 | 42 |
| 5 | 3361 | 9864 | 14 | 102 | 23 |
| 6 | 1777 | 4569 | 15 | 78 | 12 |
| 7 | 1059 | 2199 | 16 | 51 | 7 |
| 8 | 652 | 1089 | $\geq 17$ | 407 | 8 |

Table 4.14: Generalized Poisson Fit for Peaks

still clearly a rejection.

Finally, we try the generalized Poisson. The parameters for this fitting (found using the method of moments) are $\lambda = 0.1236633$ and $\phi = 3.426731$. For a chi-square test, we get 119780.7, which indicates a better fit then any of the others. Following a similar analysis, we find that the generalized Poisson is the best fit for the background as well. As such, we change the method to fit a generalized Poisson, and not a Poisson distribution, for the background and peak models. This introduces two new parameters, $\phi_0$ and $\phi_1$ which control the over-dispersion of the generalized Poisson.

We fit a Generalized Poisson model, allowing to run to convergence. It took a long time, a total of 15 iterations, but ignoring the $\phi$'s, it did eventually converge. Below is a sub-set

| Iter | $\mu_0$ | $\mu_1$ | $\phi_0$ | $\phi_1$ | $\lambda_0$ | $\lambda_1$ |
|---|---|---|---|---|---|---|
| 1 | 0.0015 | 0.1016 | 0.0 | 8.071 | 0.9947 | 0.9480 |
| 2 | 0.0018 | 0.0948 | 104.1 | 4.005 | 0.9998 | 0.5000 |
| 3 | 0.0020 | 0.7195 | 103.6 | 1.036 | 0.9999 | 0.8908 |
| 4 | 0.0019 | 0.3287 | 102.2 | 1.849 | 0.9999 | 0.9516 |
| 5 | 0.0019 | 0.1842 | 101.0 | 3.417 | 0.9999 | 0.9746 |
| 10 | 0.0018 | 0.0503 | 93.91 | 8.010 | 0.9999 | 0.9954 |
| 14 | 0.0018 | 0.0407 | 95.01 | 9.046 | 0.9999 | 0.9963 |
| 15 | 0.0018 | 0.0400 | 94.74 | 9.178 | 0.9999 | 0.9964 |

Table 4.15: Iterations for Generalized Poisson

of the iterations:

As you can see, at iteration 15, the $\mu$s and $\lambda$'s have converged. We can run a comparison to see the number of and average length of the peaks. As you can see the peaks are a lot longer, and the amount of area covered is high. We can compare to the other methods used in the HPeak paper [12]. The Generalized Poisson method is much more in line with the other methods then the Poisson method in the previous section. Mosaics, HPeak-2.1, and ChIPseeqer in the table below were run using the code provided by the two program authors, and are not from the HPeak paper.

| Method | Number of Peaks | Covered space (kb) | Avg Peak width (bp) |
|---|---|---|---|
| MACS | 22,402 | 16,940 | 756 |
| FindPeaks | 41,127 | 46,781 | 1,137 |
| SISSRs | 9,561 | 455 | 10,012 |
| CisGenome | 38,878 | 10,012 | 258 |
| Mosaics | 18,833 | 10763 | 546 |
| HPeak-b | 43,443 | 15,354 | 353 |
| Gen Pois | 34,804 | 32,941 | 958 |
| Gen Pois Matched | 10,500 | 13,016 | 1239 |
| ChIPseeqer (control) | 10,102 | 2,236 | 221 |
| HPeak-2.1 | 14,880 | 6,937 | 466 |

Table 4.16: Stat1 Results Table for Generalized Poisson

We can look at these results more closely, and compare to HPeak and ChIPseeqer. We

| | Gen Pois HiDe-Peak | HPeak | Intersection (GP) | Intersection (HP) |
|---|---|---|---|---|
| Peaks | 34,804 | 14,880 | 13,137 | 14,235 |
| Poisitons (kb) | 32,941 | 6,937 | 6,855 | 6,855 |
| | Gen Pois | ChIPseeqer | Intersection (GP) | Intersection (HP) |
| Peaks | 34,804 | 10,102 | 8,570 | 9,339 |
| Poisitons (kb) | 32,941 | 2,246 | 2,142 | 2,142 |
| | Gen Pois HiDe-Peak Matched | | HPeak | Intersection (HP) |
| Peaks | 10,500 | | 14,880 | 7,256 |
| Poisitons (kb) | 13,016 | | 6,937 | 3,472 |
| | Gen Pois Matched | | ChIPseeqer | Intersection (CS) |
| Peaks | 10,500 | | 10,102 | 4,508 |
| Poisitons (kb) | 13,016 | | 2,246 | 1,064 |

Table 4.17: HPeak and ChIPseeqer Comparisons on Stat1

can look at both the number of peaks that overlap between the two methods as well as the number of positions (in thousands). The Generalized Poisson HiDe-Peak finds most of the same peaks as HPeak and ChIPseeqer, which gives additional credibility to the method. When restricting to matched peaks only, it finds less of an overlap. Whether the positions eliminated are false peaks found by HPeak and ChIPseeqer or true peaks being eliminated inappropriately cannot be known. Later simulations suggest that restricting to matched peaks reduces error but also causes the method to miss true peaks.

## 4.10 Binned Data

The vast majority of methods bin the data at some point in the analysis. Binning the data stabilizes the distribution, so that there are not so many 0s and 1s and increasing the frequencies of higher values, making estimation easier. In addition, binning the data makes the data set smaller, giving significant savings for computational time, which is an important consideration. Each iteration of the Generalized Poisson model above took over an hour - reducing the model with bins of even size 20 makes these iterations take only minutes.

| Iter | $\mu_0$ | $\mu_1$ | $\phi_0$ | $\phi_1$ | $\lambda_0$ | $\lambda_1$ |
|---|---|---|---|---|---|---|
| 0 | 0.03658 | 0.8076 | 0 | 0 | 0.9 | 0.85 |
| 1 | 0.03467 | 0.5424 | 7.489 | 2.525 | 0.9958 | 0.7071 |
| 5 | 0.03065 | 0.1273 | 5.444 | 8.250 | 0.9988 | 0.9823 |
| 8 | 0.02419 | 0.0487 | 16.41 | 13.18 | 0.9958 | 0.9931 |
| 9 | 0.02492 | 0.0470 | 15.16 | 14.00 | 0.9957 | 0.9931 |

Table 4.18: Convergence Iterations for Binned Generalized Poisson

Therefore, the code was adjusted to allow the user to set a bin width. This requires changing the model, such that it now models only the read starts, and not the overall count at any position, so that only the second half of the joint probability - the Generalized Poisson part - is being used. Looking only at read starts removes a level of dependence, so that each count now depends only on the peak or non-peak status of the hidden chain. However, looking on read starts does not remove any information, as the counts at each position depend on the number of read starts in the preceding positions.

For a test, we chose a bin size of 20, summing together the number of read starts in non-ovelapping bins of size 20. As the majority of the bins appeared to be a zero or a one, using the cut-off method to determine initial bins seemed like the wrong choice. Instead, of initializing peak or non-peak regions, we started by selecting parameters based loosely on the parameters found for the Generalized Poisson method above. We multiplied the means of the two Generalized Poissons by 20, reduced them to Poisson by setting $\phi_0$ and $\phi_1$ to 0, and chose appropriate transition parameters. Below, you can see the convergence of the model for selected iterations.

Once the model has converged, the median peak length (before any post-processing) is 101. Given that we've used a bin size of 20, that makes the peaks length 2020. We find very few peaks with these results - 2,219 total. But they are absurdly long - average length 1,182,285. So this isn't very helpful. This indicates that there is some over-sensitivity in the

| Iter | $\mu_0$ | $\mu_1$ | $\phi_0$ | $\phi_1$ | $\lambda_0$ | $\lambda_1$ |
|---|---|---|---|---|---|---|
| 0 | 0.036581 | 0.807629 | 0 | 0 | 0.9 | 0.85 |
| 1 | 0.0346732 | 0.5424152 | 7.4899754 | 2.5259845 | 0.9958950 | 0.7071068 |
| 0 | 0.01 | 1 | 0 | 0 | 0.99 | 0.7 |
| 1 | 0.0348044 | 0.5524012 | 7.4633399 | 2.5023777 | 0.9958987 | 0.7071068 |
| 0 | 0.01 | 2 | 0 | 0 | 0.99 | 0.7 |
| 1 | 0.0360956 | 0.6123324 | 7.8600510 | 2.4406631 | 0.9959157 | 0.7071068 |

Table 4.19: Initial Value Test for Binned GP

model, creating peak regions that are much too long, that needs to be accounted for.

For a quick check in regards to initial values, we ran the algorithm with different initial values (see table below), and after just one iteration, it converges to very similar values as in the first case. The first run is the original initial values from the run above, the other two runs increase the average number of read starts in peaks ($\mu_1$) to 1 and to 2. So the algorithm is not particularly sensitive to initial values.

The question of binning suggests ask what size bin is best. In particular, how does the distribution of read starts change as we go from no bins to a bin of size 80? The table below shows the effect in regards to chromosome 1. The vast majority of bins, regardless of bin size, have 0 reads starting in them. Even when bins get up to length 80, 97% of the bins are of size 0 or 1. This seems to imply that reads only rarely start within 80 positions of each other, which helps explain the trouble the method has with identifying peak regions.

We assumed that as bins were made, the number of bins with a 'large' number of reads would increase substantially. Excluding zeros, we have the following table. Still, even with bins of length 80, 80% of the non-zero bins have only one read in them. This means that the majority of reads are isolated, a long distance away from other reads. This may be part of the problem.

Binning is important for computational reasons. Though it does not appear to work

| Bin Size | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1 | 0.99774 | 0.00210 | 0.00014 | 0.00001 | 0.00000 |
| 10 | 0.97856 | 0.01930 | 0.00177 | 0.00023 | 0.00006 |
| 20 | 0.95857 | 0.03648 | 0.00398 | 0.00061 | 0.00016 |
| 30 | 0.93955 | 0.05225 | 0.00644 | 0.00109 | 0.00030 |
| 40 | 0.92133 | 0.06685 | 0.00911 | 0.00165 | 0.00047 |
| 50 | 0.90381 | 0.08045 | 0.01197 | 0.00227 | 0.00068 |
| 60 | 0.88690 | 0.09319 | 0.01491 | 0.00301 | 0.00089 |
| 70 | 0.87053 | 0.10515 | 0.01804 | 0.00371 | 0.00115 |
| 80 | 0.85468 | 0.11644 | 0.02116 | 0.00457 | 0.00141 |

Table 4.20: Bin Distribution

| Bin Size | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0.9282 | 0.0639 | 0.0058 | 0.0010 | 0.0003 | 0.0002 | 0.0001 |
| 10 | 0.9002 | 0.0827 | 0.0109 | 0.0027 | 0.0011 | 0.0006 | 0.0004 |
| 20 | 0.8806 | 0.0960 | 0.0146 | 0.0038 | 0.0016 | 0.0009 | 0.0005 |
| 30 | 0.8643 | 0.1065 | 0.0181 | 0.0049 | 0.0020 | 0.0011 | 0.0007 |
| 40 | 0.8498 | 0.1158 | 0.0210 | 0.0060 | 0.0025 | 0.0013 | 0.0008 |
| 50 | 0.8363 | 0.1245 | 0.0236 | 0.0071 | 0.0029 | 0.0015 | 0.0009 |
| 60 | 0.8240 | 0.1318 | 0.0266 | 0.0079 | 0.0034 | 0.0018 | 0.0011 |
| 70 | 0.8122 | 0.1394 | 0.0286 | 0.0088 | 0.0038 | 0.0021 | 0.0012 |
| 80 | 0.8013 | 0.1456 | 0.0315 | 0.0097 | 0.0042 | 0.0022 | 0.0014 |

Table 4.21: Bin Distribution Excluding Zeros

well here, due to some over-sensitivity, the models presented later use binning and remain accurate. Part of the issue is that peak and background regions are being selected based only on the number of reads in any bin. Giving the model additional information, like control data, GC Content, and mappability bias, helps the model to make better decisions in determining peak locations.

CHAPTER 5

# CONTROL DATA AND COVARIATES

The models presented so far examine only the sequence of DNA fragment counts, without taking into account any possible outside factors or known DNA properties. In this chapter, we develop a model that accounts for control data as well as several other covariates known to have an impact of the DNA fragment count at any given position.

Up to this point, the model has focused on accounting for the heavily correlated nature of the counts. Currently, the number of DNA fragments at each position is recorded, and, as each fragment is 250 bps in length, a natural correlation is induced among consecutive positions. Instead, the data can be recorded differently, with each fragment recorded only at the position at which it begins. In that case, the counts at each position would be independent, and the resulting hidden Markov model would resemble that used by HPeak, with no extra dependence among the counts except that induced by the hidden Markov chain.

The result of modeling on the fragment starting points is that the data indicates only the beginning of each peak on the forward strand and the end of each peak on the reverse strand. Therefore, the resulting peak regions would have to be extended by the average read length, and the information on both strands combined in a post-processing step, to fully identify peak regions. Restricting the 'found' peaks to only those that appear on both strands (called 'matched' peaks) helps eliminate some false positives, and the center of such peaks would be the actual binding site.

Once the data is restricted to only fragment starting positions, the model is simplified, and it now easier to let $X_i$ depend on the hidden state as well as any number of possible

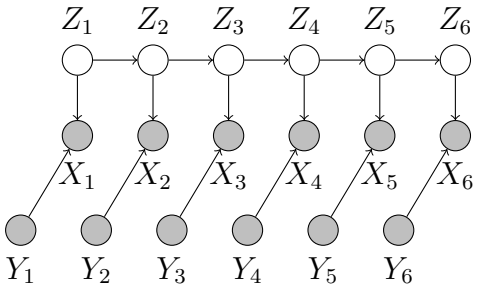


Figure 5.1: Hidden Markov Model with Covariates

covariates. The model is thus:

$$\Pr(X_i|Z_i) \sim \text{Poisson}(\mu_i)$$

where $\mu_i$ is a function of covariates (Y). That function would be different depending on whether or not $Z_i$ is a 1 on 0:

$$\log(\mu_i)|Z_i = \beta_{Z_i} Y_i$$

Graphically, the model resembles figure 5.1.

This model is a special case of an input-output HMM (Bengio and Frasconi, 1995 [1]). The input-output HMM also assumes that the observed covariates $Y$ can affect the value of the hidden states $Z$ - removing this assumption just simplifies the model. The authors of the referenced paper describe an E-M algorithm to estimate the parameters, but as this relies on a forward-backward algorithm, its computational complexity is too high for our setting. However, this can be coded in the same way as the base case, by using a Viterbi algorithm, and then estimating the covariate functions for each group - the peak and non-peak.

## 5.1 Control

Most recent ChIP-seq experiments include a 'control' or 'input' lane, where no immunoprecipitation is performed, such that the data contains all DNA fragments, regardless of the presence or absence of the transcription factor of interest. In theory, the background ought to be uniform, if all positions on the DNA sequence were equally likely to be the start of the broken DNA fragments. In fact, however, the background is not uniform, due to the structure and nature of DNA and the experimental process. Kharchenko et al (2007) [7] point out that the background has three major characteristics that vary from the expected uniform: wide regions of high counts, specific lone positions with very high counts, and regions with high counts resembling peaks. Having a control, and taking its values into account, can help deal with these issues. If the control is ignored, the method is much more likely to pick up these false background peaks.

In order to include the control data in the model, it is important to examine the relationship between control data and the experimental data. Using the peaks determined by the generalized Poisson version of HiDE-peak on the Stat1 dataset, we can calculate the average count value for each possible recorded control value. Below are the images, restricting control values to below 100 and then to below 50. Very rarely control data increases up to as much as 400, but these points are left out of the images below, as the vast majority of control counts are below 50.

Note that the relationship between the average count and the control is relatively linear. The slope of the two lines is .352 for the background regions and .4465 for the peak regions, showing that the relationship is stronger for peak regions than for background regions. The variance increases with the control, but the vast majority (99.9%) of the positions have a control value of less than 20. Histograms for the control are omitted here because the vast

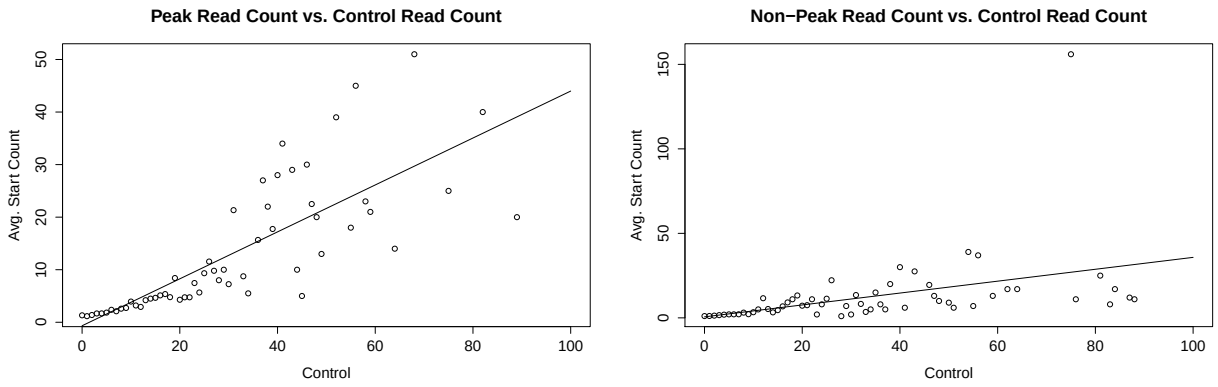


Figure 5.2: Control Counts below 100 for Peak and Non-Peak

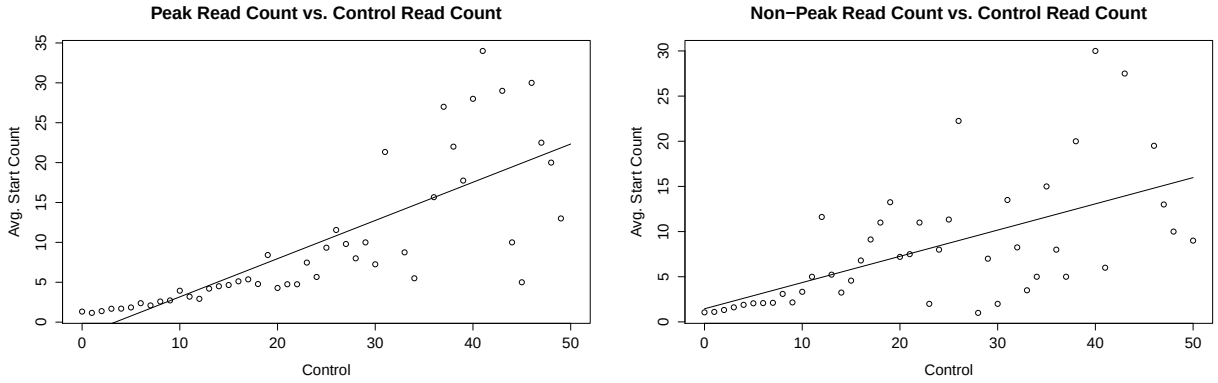


Figure 5.3: Control Counts below 50 for Peak and Non-Peak

majority of the control (97%) is 0 for both peak and non-peak.

Because we use a log link in the Poisson regression, we are interested in the relationship between the log average counts and the control. Doing so (as in the images below) removes the linear relationship. This can be accounted for by transforming the control, as well. Taking the the square root of the control appears to create a linear relationship, as can be seen in figure 5.6.

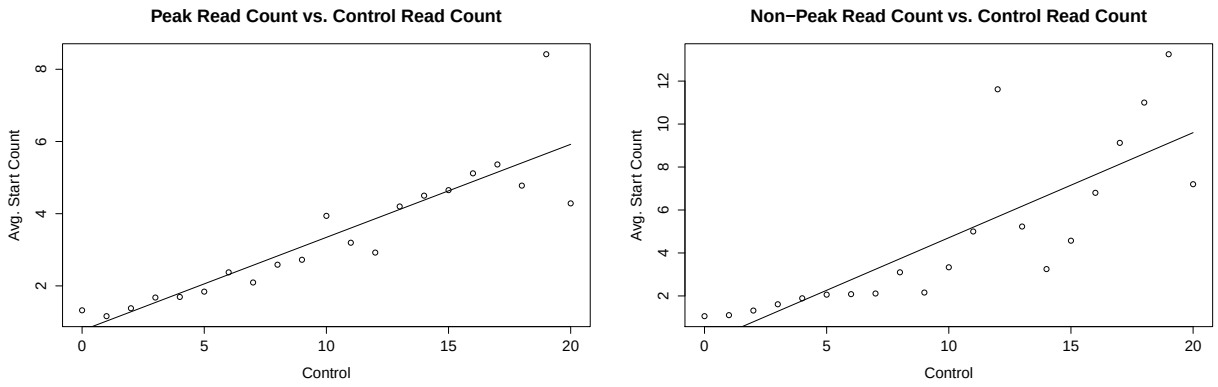


Figure 5.4: Control Counts below 20for Peak and Non-Peak

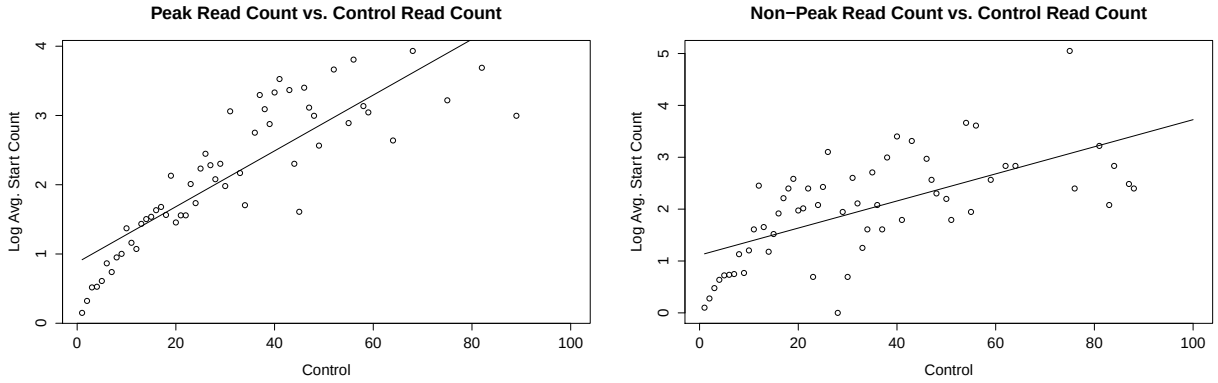


Figure 5.5: Log Average Counts for Peak and Non-Peak

## 5.2 GC Content

Kuan et al. (2011) [11] identify GC Content as a key covariate influencing DNA fragment behavior. The hope is that by taking GC content into account, model accuracy will improve. Using the peak regions determined from the Generalized Poisson version of HiDe-Peak, we can examine the behavior of the GC Content for both peak regions and non-peak regions. The GC Content file available calculates the average number of G or C bases across a read length of 125 starting at each position of the genome. It then further averages those values

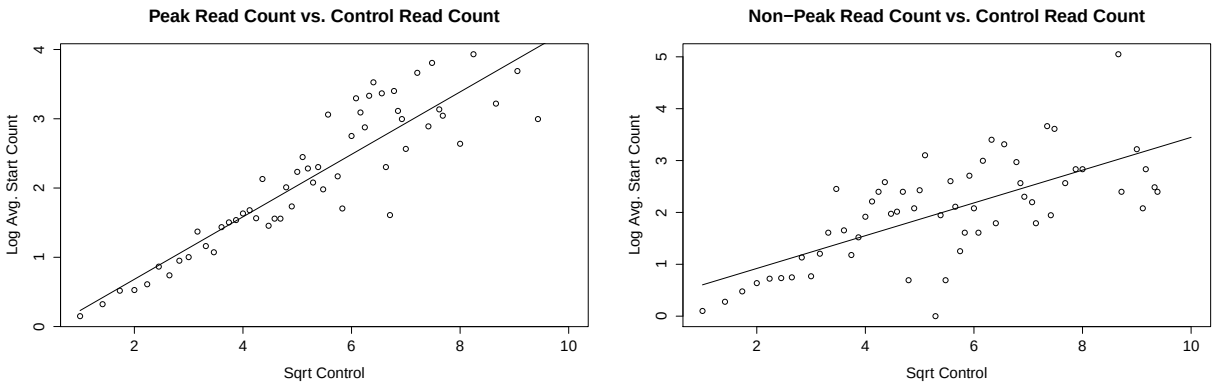


Figure 5.6: Log Average Counts with Square Root Control for Peak and Non-Peak

across every 200 positions, and lists only this GC average every 200th base pair.

To examine the relationship between GC content and average read count, GC Content was rounded to the nearest hundredth and treated as having 100 categories. For each possible value of GC Content rounded to the 100th, the number of read starts for peak and non-peak regions was calculated. Because only every 200th was available, each position used the nearest 200th GC content. From figure 5.7, it can be seen that the average read count for peak and non-peak behave quite differently. In addition, the histogram of GC Content appears to be different for peak and non-peak regions. The mean GC content for peak is .478, whereas the mean GC Content for Non-Peak is .428. This is not a substantial difference, but it is significant. Note also that the relationship between GC Content and average read starts is not linear for the Peak regions, which makes estimating its effect more challenging.

Because the model uses a log link function for the Poisson regression, figure 5.8 below show the relationship between GC content and log average count. The quadratic relationship for the peak regions appears to hold, so no further transformation is required.

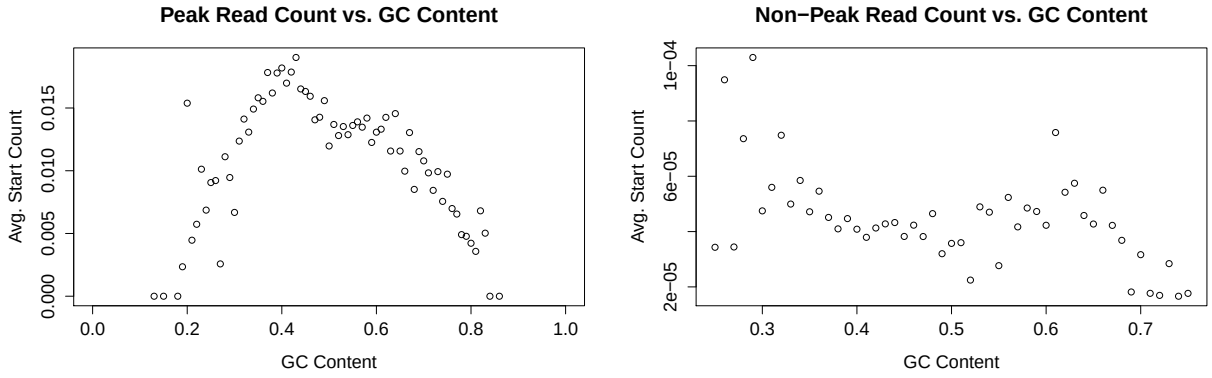


Figure 5.7: Peak Average Count for GC Level

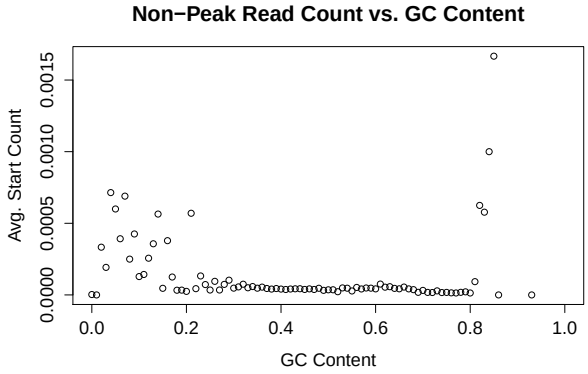


Figure 5.8: Non-Peak Average Count for GC Level

## 5.3 Mappability

The Mosaics paper by Kuan et. al (2011) [11] mentions three covariates - Control, GC Content, and Mappability. When not binning, Mappability is merely a 0-1 variable, indicating whether it is possible for a read of length $n$ to map uniquely to that position. Much of the genome consists of repeats, and as much as 20% of the genome may consist of un-mappable positions. The algorithm used by the high-throughput machines simply discards any reads that cannot be uniquely aligned, thereby forcing all un-mappable positions to have a count

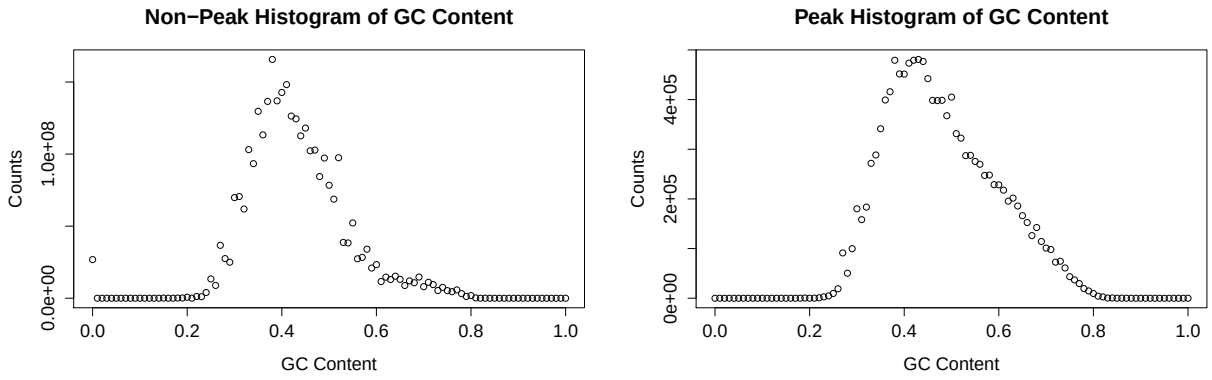


Figure 5.9: GC Content Histogram for Peak and Non-Peak

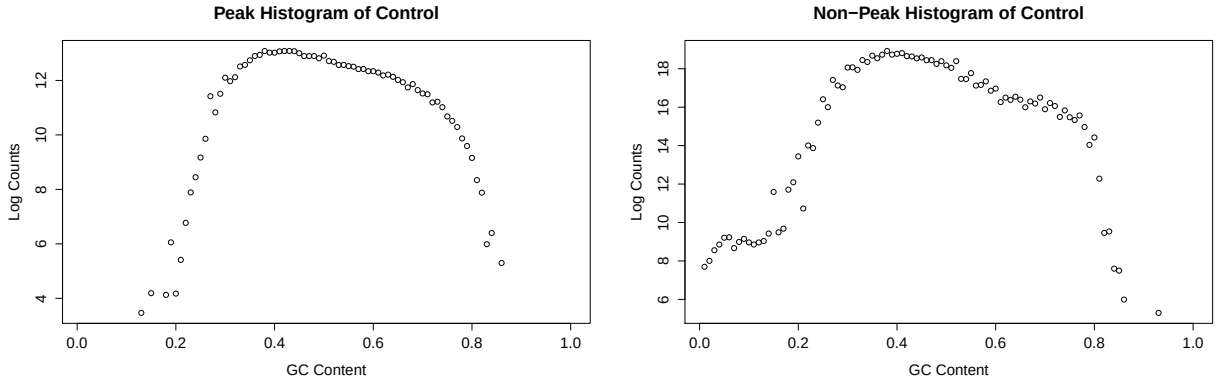


Figure 5.10: Average Log Count for GC Content

of 0. Naturally, knowing this information is valuable when trying to identify peaks.

For Kuan et al, who look at the number of reads overlapping a position, and not just the reads starting at a position, the mappability factor of any given position is dependent not only on whether a read can map uniquely to that position, but also whether a read can map uniquely to any of the positions $n$ previous, as reads starting at those positions would overlap the position of interest. Therefore, the mappability factor of any position is the average of the 0-1 indications of mappability for itself and all positions $n$ previous. Because Kuan et al bin, the mappability factor for each bin is the average of the mappability factors

across all positions within a bin. This information, for bins of size 200, is readily available, and as such, is what we currently use for mappability.

We want to examine the effect of mappability on peak and background areas. Kuan et. al suggest looking at $\log(M+1)$ (labeled 'log mappability' in the plots in figures 5.11 and 5.12). For the peak regions, it appears that the peak signal overcomes the effect of mappability. Two lines are plotted on the peak regions in figure 5.11- the flat one is a line at the mean, the other is a fit line to the points. As these are very similar, it appears that there is no need to include mappability as a factor in the peak regions. For the background regions, however, there is clearly a relationship between mappability and average count in the background regions. The line fit uses the square root of $\log(M+1)$, and fits very well.

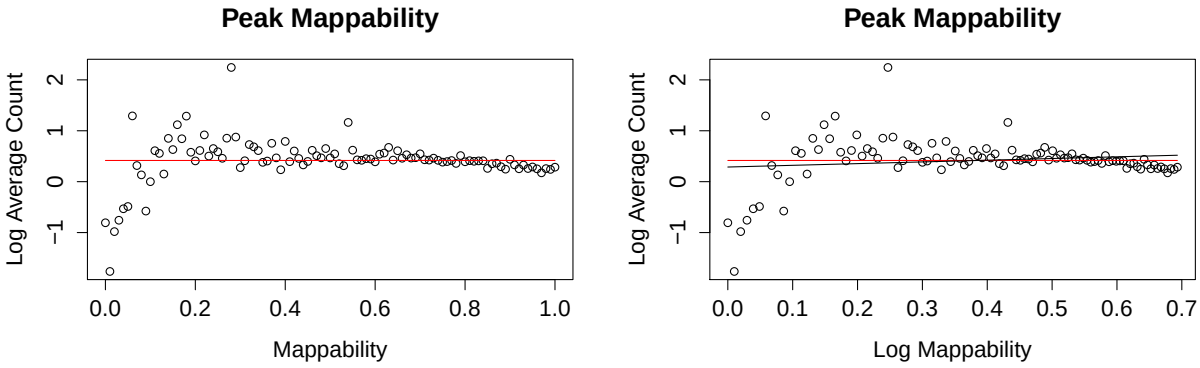


Figure 5.11: Mappability for Peak

It's interesting to note the relationship mappability has with other covariates. Below are plots for Mappability against GC Content and the Control. The control has a very similar relationship to mappability to that of the background regions. GC content has a clear, but rather complicated relationship, as well.

Adding mappability as a covariate to the model, however, creates an unstable IRLS algorithm for the Poisson regression that will not converge. The condition number for the

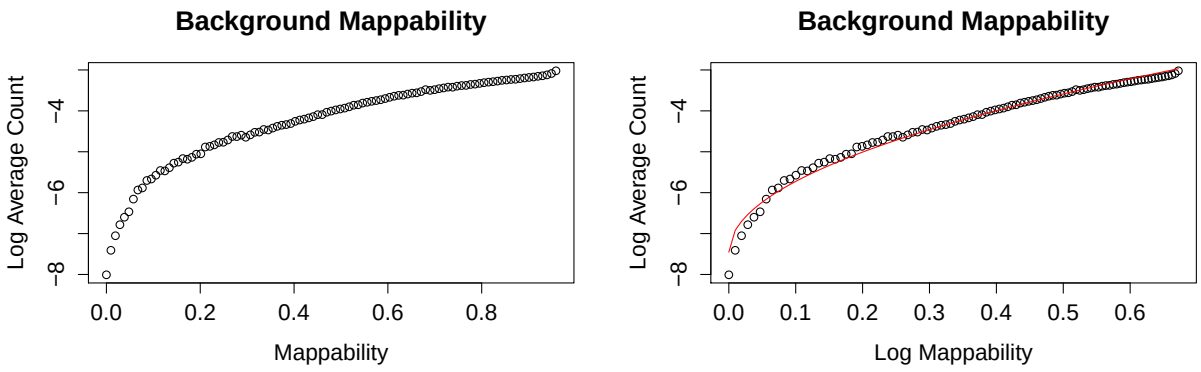


Figure 5.12: Mappability for Background

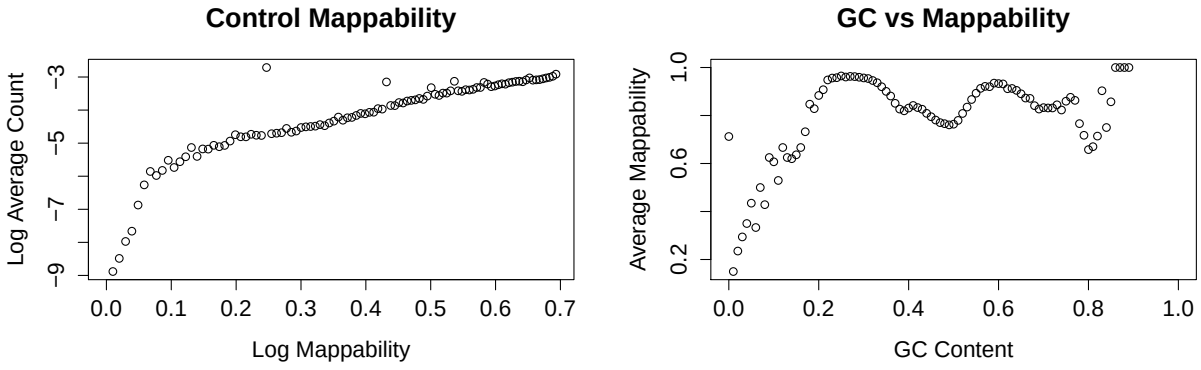


Figure 5.13: Mappability vs. Control and GC Content

second derivative matrix used in the IRLS algorithm for the fit model for chromosome 1 in R is over 200, indicating an unstable matrix. The histogram for mappability shows that mappability is nearly binary with 32.3% of the positions exactly 0, meaning no reads can map there, and 25.8% exactly 1. In fact, nearly 40% of the reads are above .9.

As a result of this extreme bi-modality, we treat mappability as a 0-1 variable. When calculating the Viterbi algorthm, in the case when when mappability is zero, we must set $p_0$ and $p_1$, the probability of the the position being background or peak, respectively. If $p_0 = p_1$, then the algorithm will just remain in peak or background depending on the

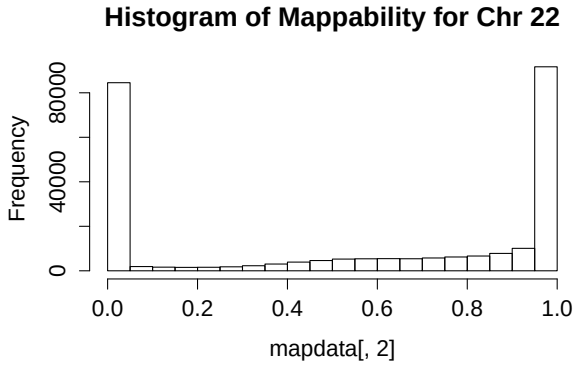


Figure 5.14: Histogram of Mappability

previous positions state, as it has no evidence to change. If instead, we set $p_0 = 1$ and $p_1 = 0$, then the algorithm is strongly encouraged to switch to background on finding a position with mappability 0. As it is unlikely for there to be a peak in a mappability 0 location, we set $p_0 = 1$ and $p_1 = 0$.

In the case when mappability is non-zero, we have the following model:

$$Y_i|Z_i \sim \text{Poisson}\left(\lambda_{i,Z_i}\right)$$

$$\log(\lambda_{i,Z_i}) = \beta_{0,Z_i} + \beta_{1,Z_i}\sqrt{(C_i)} + \beta_{2,Z_i}GC_i I_{.2<GC_i<.8} + \beta_{3,Z_i}GC_i^2 I_{.2<GC_i<.8}$$

where $\beta_{2,0} = \beta_{3,0} = 0$, as GC content has no effect on background.

When mappability is zero, $Y_i|Z_i = 0$. As no reads can map to any position with mappability zero, it is known that $Y_i$ is zero for all cases, regardless of the value of $Z_i$. We can model this with an iterative Viterbi algorithm just as with the simpler model.

## 5.4 Negative Binomial

When this Poisson model is fit to the Stat1 dataset, the average squared Pearson residual for the peak regions is above 3, indicating over-dispersion. As such, it makes sense to use Negative Binomial regression instead of Poisson regression for the peak regions. We thus have the following model when mappability is non-zero:

$$\Pr(Y_i|Z_i) = \begin{cases} \text{Pois}(\lambda_i) & \text{if } Z_i = 0 \\ \text{NegBi}(\mu_i, \alpha) & \text{if } Z_i \neq 0 \end{cases}$$

where

$$\log(\lambda_i) = \beta_{0,0} + \beta 1, 0\sqrt{C_i}$$

$$\log(\mu_i) = \beta_{0,1} + \beta_{1,1}\sqrt{C_i} + \beta_{1,2}GC_i + \beta_{1,3}GC_i^2$$

where $C_i$ is the control value at position $i$ and $GC_i$ is the GC content at position $i$.

The Negative Binomial distribution has two parameters, the mean $\mu$ and the index $\alpha$, which controls the amount of over-dispersion. Typically, fitting a Negative Binomial is a two-step process, using an IRLS Process to estimate the linear predictor $\log(\mu_i)$, and then a line-search algorithm over the log-likelihood to find $\alpha$. To save time, we estimate $\alpha$ from the Poisson fit, and leave it fixed, using only the IRLS algorithm to find the linear predictor parameters. An estimate of $\alpha$ is found using the following relationship:

$$\hat{\phi} = 1 + \frac{\bar{y}}{\alpha}$$

where $\bar{y}$ is the average fitted value from the Poisson regression and $\hat{\phi}$ is the Pearson estimate of the dispersion from the Poisson model fit.

## 5.5 Data Simulations

Simulated data was needed such that the peaks depended on GC content, control, and mappability. The earlier simulations assigned reads randomly. Rather than generating new GC content, control, or mappability, the real GC content, mappability profile, and control data for the Stat1 data set was used. Peaks were assigned at random, and then a weight was calculated for each position, depending on its peak or background status, and its associated mappablity, GC content, and control value. To generate additional variability, a random weight was assigned to each peak, and the weight for each position in that peak region was adjusted accordingly. The reads were then assigned to the positions proportionally to the calculated weights.

The simulation algorithm takes the following inputs:

- $l = 12362487$: the length of the 'dataset'
- $M = 3584$: number of actual peaks
- $N_c = 986681$: Number of control reads
- $N = 1248474$: Number of reads
- $k = 0.05$ :% of reads found in peaks
- $rl = 10$: Read length (accounting for Bin size)
- $bin = 20$: Bin size
- Parameters:
  - $B_peak$: Parameters for the GC and Control Effect on Peaks
  - $B_background$: Parameters for the control effect on the background

The algorithm then follows several steps:

- Step 1: Decide on positions to be looked at, and get GC Content
  - Select peak regions randomly
  - Define the peak regions around the peaks
  - Find (and record) the GC Content of each position for control and in each peak region
- Step 3: Background
  - Find number of background reads - $N * (1 - k)$
  - Find weight of each background position
  - Assign background reads according to weights
- Step 4: Peak regions
  - For each peak, assign an extra weight, $v_i$, to account for over-dispersion
  - For each peak position, calculate the total weight, based on control, $v_i$, and GC content
  - Assign peak region reads based on weights
- Step 5: Record output

The simulation algorithm shares peaks across forward and reverse reads on the same chromosome, and then assigns reads separately. The output is in mock ELAND format, so that it can be read by other existing programs, like ChIPseeqer, HPeak and Mosaics.

The first simulation assumed that the control depended only on GC content, and so had no 'sticky' positions, extra enriched areas, or false peaks. This made peaks easy to locate, and also reduced the impact of GC content on the model outputs.

Something to examine is the peak elimination step at the end of the process, which removes all peaks that do not have a symmetrical matched peak on the opposing strand. This is a method used by a variety of the papers, including Jothi [6], Kharchenko [7], and Zhang [16]. Table 5.1 includes all possible results - Poisson and Negative Binomial both with and without removing unmatched peaks. Additionally, the method is run on HPeak for comparison.

| | Peaks Found | Avg. Peak Length | Peaks Sim | # Peaks Missed | # False Peaks | # Correct Peaks |
|---|---|---|---|---|---|---|
| Pois Matched | 11949 | 407 | 28871 | 16836 | 4 | 12035 |
| Pois Un-Matched | 29665 | 457 | 28871 | 923 | 1850 | 27948 |
| NegBi Matched | 9489 | 438 | 28871 | 19307 | 2 | 9564 |
| NegBi Un-Matched | 26466 | 464 | 28871 | 2418 | 171 | 26453 |
| HPeak | 25886 | 310 | 28871 | 2876 | 6 | 25995 |

Table 5.1: Simulation 1 Results

Note that allowing unmatched peaks increases the errors, but also significantly increases the number of correctly identified peaks. The Negative Binomial without removing unmatched peaks appears to be the only method to outperform HPeak. Part of this may be due to the way that the data is simulated. To allow for over dispersion, each peak was given an additional random weight. This was done independently on each strand, so that a strong peak on one strand can be a weak one on the other. Since this method appears to miss weaker peaks, eliminating unmatched peaks eliminates peaks of this form, causing many more peaks to be missed.

The second dataset did not simulate the control, but instead used the real control data from the Stat1 dataset, thus guaranteeing the presence of non-uniformities in the control

| | Peaks Found | Avg. Peak Length | Peaks Sim | # Peaks Missed | # False Peaks | # Correct Peaks |
|---|---|---|---|---|---|---|
| Pois Matched | 4208 | 440 | 28871 | 24668 | 3 | 4203 |
| Pois Un-Matched | 26150 | 417 | 28871 | 3445 | 798 | 25426 |
| NegBi Matched | 4150 | 450 | 28871 | 24719 | 1 | 4152 |
| NegBi Un-Matched | 26357 | 434 | 28871 | 3234 | 803 | 25637 |
| HPeak | 21670 | 307 | 28871 | 7187 | 47 | 21684 |
| Mosaics | 17647 | 458 | 28871 | 11123 | 5 | 17748 |

Table 5.2: Simulation 2 Results

data. From there, the peak and background reads were simulated the same way as in the previous example with one change. Before, the random 'peak strength' that was added to include extra variability was simulated with a Gaussian with mean 0 and variance 4. Now, a log-gamma distribution is used, so as to simulate the data closer to the negative binomial.

The parameter estimates show a stronger signal for the control now, though GC content still behaves quite differently. Once again, the best performing method is the negative binomial regression when unmatched peaks are not removed. Though the dataset was simulated with an intended over-dispersion of 6, the Poisson regression model identifies a $\hat{\phi}$ of only 3.9, and the resulting $\alpha$ for the negative binomial is $\alpha = 1.192508$.

The results below show that, once again, it is best to leave in unmatched peaks. With this noisier control, the negative binomial and Poisson results are much closer, though the negative binomial still outperforms the Poisson regression. When compared to HPeak, the negative binomial unmatched finds many more (nearly 4000) of the peaks, but does have a larger number of false peaks. Mosaics, the method that accounts for covariates, but not the dependence in the data, has very few false positives, but misses a large number of peaks in comparison to the other methods.

ChIPseeqer gives a segmentation fault partway through compiling, and so it is impossible to compare full results to ChIPseeqer. However, ChIPseeqer does return peaks for a small

| | Peaks Found | Avg. Peak Length | Peaks Sim | # Peaks Missed | # False Peaks | # Correct Peaks |
|---|---|---|---|---|---|---|
| Pois Matched | 1594 | 461 | 10060 | 8470 | 2 | 1590 |
| Pois Un-Matched | 9205 | 420 | 10060 | 1168 | 340 | 8892 |
| NegBi Matched | 1566 | 470 | 10060 | 8493 | 1 | 1567 |
| NegBi Un-Matched | 9269 | 436 | 10060 | 1105 | 343 | 8955 |
| HPeak | 7685 | 309 | 10060 | 2370 | 22 | 7690 |
| Mosaics | 6317 | 459 | 10060 | 3705 | 3 | 6355 |
| ChIPseeqer | 2775 | 199 | 10060 | 8481 | 1190 | 1579 |

Table 5.3: Simulation 2 Results with ChIPseeqer

number of chromosomes (1 and 10-16) before crashing, and as such we can compare the results for those results to ChIPseeqer. Table 5.3 shows the results with ChIPseeqer accounting for the control, using the default cut-off. It can be clearly seen that ChIPseeqer is much too conservative. While dropping the cutoff will increase the number of correct peaks found, it will also increase the already high number of false peaks.

The next two simulations change the parameter values used above, making the effect of the covariates weaker. The first simulation uses a gamma $(\frac{1}{6}, 6)$ to create the over-dispersion, thereby creating a small number of very strong peaks. Many more peaks are missed by all of the methods in this case, though HPeak performs best, capturing the largest number of peaks. Simulation 4 significantly reduces the over-dispersion, from 6 to 2. HPeak still outperforms the other methods, though by less of a large margin. Because the strength of HiDe-Peak and Mosaics lies in its ability to account for covariate effects, reducing the covariate effects in the simulated data reduces their efficacy, and in this case, HPeak is the appropriate method.

The final simulation makes the effect of GC content very strong, while weakening the over-dispersion. In this case, HPeak performs worse, as it does take the covariate information into account. Mosaics performs best in this case, as there is little over-dispersion ($\phi = 1.2$)

| | Peaks Found | Avg. Peak Length | Peaks Sim | # Peaks Missed | # False Peaks | # Correct Peaks |
|---|---|---|---|---|---|---|
| Pois Matched | 2936 | 406 | 28871 | 25928 | 0 | 2943 |
| Pois Un-Matched | 7865 | 435 | 28871 | 21063 | 72 | 7808 |
| NegBi Matched | 2630 | 411 | 28871 | 26233 | 0 | 2638 |
| NegBi Un-Matched | 8405 | 443 | 28871 | 20525 | 88 | 8346 |
| HPeak | 9399 | 343 | 28871 | 19499 | 27 | 9372 |
| Mosaics | 8314 | 525 | 28871 | 20532 | 1 | 8339 |

Table 5.4: Simulation 3 Results

| | Peaks Found | Avg. Peak Length | Peaks Sim | # Peaks Missed | # False Peaks | # Correct Peaks |
|---|---|---|---|---|---|---|
| Pois Matched | 5157 | 455 | 28871 | 23707 | 11 | 5164 |
| Pois Un-Matched | 14264 | 426 | 28871 | 15183 | 623 | 13688 |
| NegBi Matched | 4965 | 464 | 28871 | 23887 | 9 | 4984 |
| NegBi Un-Matched | 13991 | 439 | 28871 | 15205 | 377 | 13666 |
| HPeak | 14775 | 327 | 28871 | 14107 | 33 | 14764 |
| Mosaics | 13033 | 497 | 28871 | 15782 | 1 | 13089 |

Table 5.5: Simulation 4 Results

and GC content has a large effect on peak size. Because of the low over-dispersion, the negative binomial is not necessary, and performs comparable to the Poisson regression case. In regards to the number of true peaks found, the Poisson and negative binomial outperform HPeak, but HPeak has many fewer false peaks.

In cases where there is some over-dispersion and a noticeable covariate effect, the negative binomial hidden Markov model appears to be the best choice. However, when the covariate

| | Peaks Found | Avg. Peak Length | Peaks Sim | # Peaks Missed | # False Peaks | # Correct Peaks |
|---|---|---|---|---|---|---|
| Pois Matched | 10533 | 10126 | 25299 | 14704 | 1 | 10595 |
| Pois Un-Matched | 25162 | 9815 | 25299 | 728 | 726 | 24571 |
| NegBi Matched | 12968 | 10017 | 25299 | 12326 | 71 | 12973 |
| NegBi Un-Matched | 24706 | 9738 | 25299 | 1141 | 680 | 24158 |
| HPeak | 24066 | 6012 | 25299 | 1181 | 30 | 24118 |
| Mosaics | 24555 | 9663 | 25299 | 663 | 116 | 24636 |

Table 5.6: Simulation 5 Results

| Method | $\beta_{0,0}$ | $\beta_{C,0}$ | $\beta_{0,1}$ | $\beta_{GC,1}$ | $\beta_{GC^2,1}$ | $\beta_{C,1}$ | $\lambda_0$ | $\lambda_1$ | $\alpha$ |
|---|---|---|---|---|---|---|---|---|---|
| Poisson | -3.2293 | 0.6582 | 0.0630 | 2.2667 | -3.3610 | 0.0743 | 0.9993 | 0.7937 | |
| Negative Bin | -3.2879 | 0.6134 | -2.0880 | 7.6169 | -7.9784 | 0.3698 | 0.9994 | 0.8908 | 0.1798508 |

Table 5.7: Stat1 Parameters

effect is minimal, HPeak outperforms this method. Similarly, if there is little over-dispersion and a large covariate effect, Mosaics appears to perform best. It is also worth noting here that these are all single simulations for each case, and the results of further simulations may vary.

## 5.6 Stat1 Results

As a final check, it makes sense to compare this method on the real Stat1 data-set. The parameters found for the two methods, the Poisson and the negative binomial, do not vary much for the background, however, the GC content parameters change significantly for the peak. This has the result of finding more peaks which are also, on average, longer than the ones found by the Poisson method. The results are in table 5.7.

The results for a variety of methods, including HPeak, Mosaics, and HiDe-Peak, are in table 5.8. HiDe-Peak appears to be over-sensitive when using un-matched peaks, but when the method restricts to matched peaks, it performs much better, and, in terms of peak numbers, more in line with other methods. Restricting to matched peaks may remove true peaks only present on one strand, but it also helps eliminate a number of false positives.

Looking only at matched peaks, we may be interested in how much overlap there is among peaks found by different methods, particularly among the strongest peaks found by each method. To identify the strongest peaks for HiDe-Peak, the posterior probability of the

| Method | Number of Peaks | Covered space (kb) | Avg Peak width (bp) |
|---|---|---|---|
| MACS | 22,402 | 16,940 | 756 |
| FindPeaks | 41,127 | 46,781 | 1,137 |
| SISSRs | 9,561 | 455 | 10,012 |
| CisGenome | 38,878 | 10,012 | 258 |
| Mosaics | 18,833 | 10763 | 546 |
| HPeak-b | 43,443 | 15,354 | 353 |
| HiDe-Peak Poisson Matched | 18,066 | 9,211 | 509 |
| HiDe-Peak Poisson UnMatched | 118,312 | 32,300 | 273 |
| HiDe-Peak Neg Bi Matched | 21,507 | 20,367 | 947 |
| HiDE-Peak Neg Bi UnMatched | 106,757 | 41,008 | 384.1 |
| ChIPseeqer (control) | 10,102 | 2,236 | 221 |
| HPeak-2.1 | 14,880 | 6,937 | 466 |

Table 5.8: Stat1 Results Table

count at each position in a peak was calculated using the estimated parameters of the linear predictor. Each peak's score was calculated as the average of those probabilities across the length of the peak. Mosaics, HPeak, and ChIPseeqer also each included a score for each peak, based on the posterior probability of their models, so that it is possible to rank peaks by chromosome.

For each chromosome and for each method, the top 100 peaks as determined by the posterior probability scores were compared. Table 5.9 shows the proportion of the top 100 peaks that overlapped between each pair of methods, averaged across all chromosomes. A score of 1 means that the top 100 peaks for both methods were exactly the same, whereas as score of 0 shows that none of the top 100 peaks overlapped between the two compared methods. Figure 5.15 shows the proportion of overlap between HiDe-Peak and the other three methods by chromosome.

There is a clear variability in the proportion of overlap between the methods by chromo-

some. We can do a statistical test of the equality of these proportions:

$$H_0 : p_i = p \quad \forall i$$

versus the alternative, where at least one is different, by fitting the logistic model

$$\text{logit}(p_i) = \beta_0$$

versus the alternative model

$$\text{logit}(p_i) = \beta_0 + \beta_i$$

in which $\beta_i$ is a chromosome specific effect. The first test, for the comparison to HPeak, results in a residual deviance of 29.022, which, when compared to chi-square with 22 degrees of freedom, gives us a p-value of .17, so that we cannot reject the null. For the other comparisons, however, to ChIPseeqer and Mosaics, the residual deviances are 84.956 and 122.29, respectively. In these two cases, we reject the null, and conclude that these proportions are statistically different by chromosome. For Mosaics, this is likely due to the differing amount of control and GC content found on each chromosome, as Mosaics identifies peaks on a chromosome-by-chromosome basis. In particular, Mosaics identifies no peaks on Chromosome 9, likely due to one particularly strong background count which affects model estimation for that chromosome.

| | HPeak | ChIPseeqer | Mosaics |
|---|---|---|---|
| HiDe-Peak | 0.448 | 0.377 | 0.180 |
| HPeak | | 0.572 | 0.220 |
| ChIPseeqer | | | 0.220 |

Table 5.9: Top 100 Peak Comparisons

Figure 5.16 shows the overlap between the different methods, excluding HiDe-Peak. The table shows that while HPeak and ChIPseeqer agree on their top 100 peaks about 60% of the time, the agreement between Mosaics and these two methods is much lower.

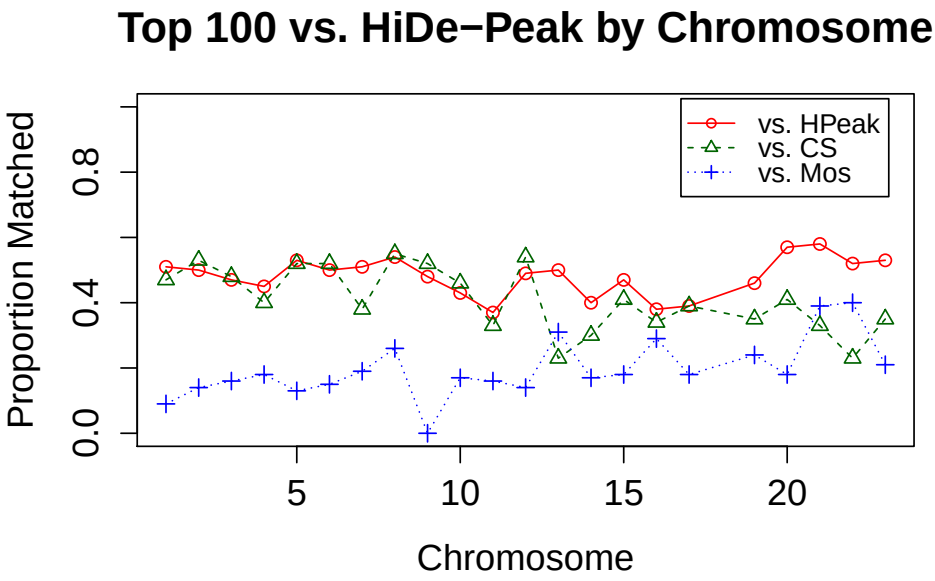


Figure 5.15: Top 100 vs. HiDe-Peak by Chromosome

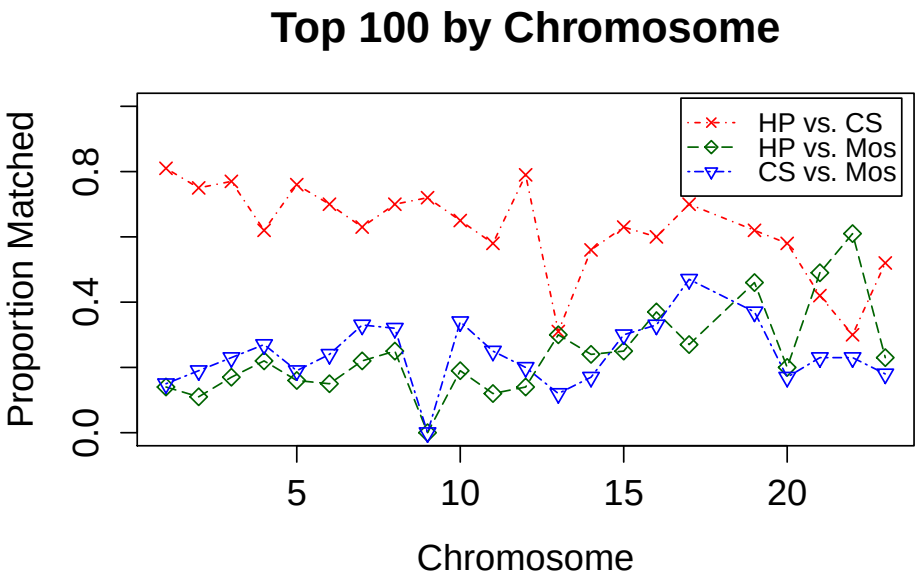


Figure 5.16: Top 100 Comparisons by Chromosome

We may also have an interest in whether the top 100 peaks for each chromosome identified by HiDe-Peak were identified at all by the other methods. Table 5.10 shows the proportion of the HiDe-Peak top 100 peaks by chromosome that were also identified by the other methods. Although these proportions are higher, figure 5.17 indicates that there are substantial differences in peaks identified by the different methods.

| | HPeak | ChIPseeqer | Mosaics |
|---|---|---|---|
| HiDe-Peak | 0.709 | 0.395 | 0.700 |

Table 5.10: Top 100 HiDe-Peak Compared to Other Methods

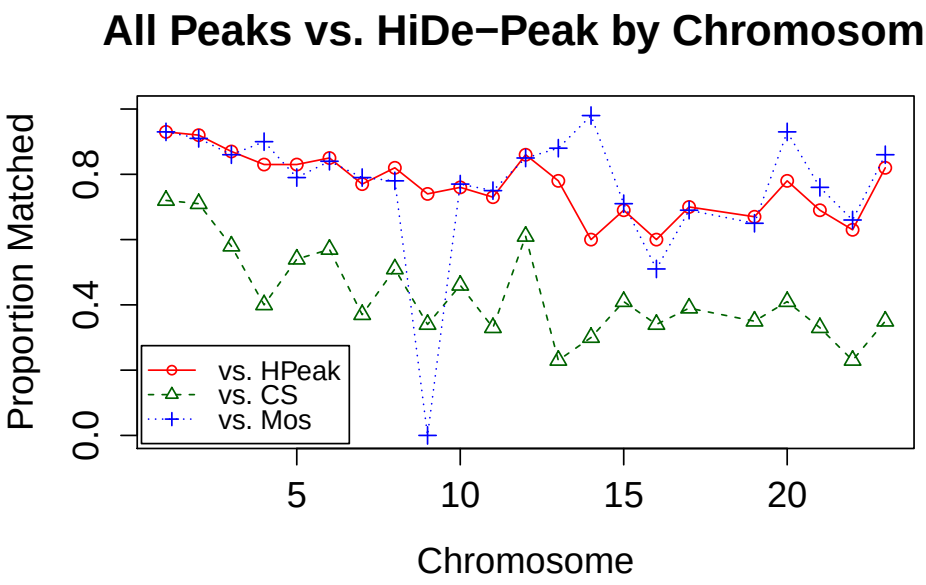


Figure 5.17: Top 100 HiDe-Peak Comparisons by Chromosome

It is surprising how many of the top 100 strongest peaks found by HiDe-Peak are not found by the other methods. To examine why this might be the case, we look at three examples of peak regions in figure 5.18, all from chromosome 1. Any peak with a max count below a certain value will not be located using ChIP-seq, so many of the more subtle peaks

are identified by HiDe-Peak and not ChIP-seq. All three peaks in figure 5.18 have low max counts, which is why none of them are identified by ChIP-seq.

The first peak in figure 5.18 was identified only by HiDe-Peak, and was not identified by either Mosaics or HPeak. The HiDe-Peak model is likely to declare a peak with a high GC content, as this peak has. In addition, though there is no position with a particularly high read count, the peak has multiple positions with low numbers of reads, and the hidden Markov model structure of HiDe-Peak allows for the model to identify regions of this type.

The second peak is identified by both HPeak and HiDe-Peak. This model has a single higher position surrounded by a long region of non-zero read counts, and, due to the hidden Markov model structure of both methods, it is identified as a peak by both HPeak and HiDe-Peak. The third peak is identified by HiDe-Peak, HPeak, and Mosaics. This peak looks like the other two peaks, with its long region of low count enrichment, but it also has a lower GC content than the other peaks. The model used for Mosaics finds that peaks are most likely with a GC content of about .6, which is approximately the GC content level for part of this peak.

These comparisons show that while many of the top peaks found by HiDe-Peak are also identified by the other methods, there is still a great deal of disagreement among the different peak finding methods. HiDe-Peak identifies peaks with higher GC content than those found by Mosaics, and like HPeak, can identify long regions of low enrichment as being peaks, not just those positions with high counts. By accounting for the control, the count dependence, and GC content, HiDe-Peak is able to identify a wider array of candidate protein binding regions then its competitors.

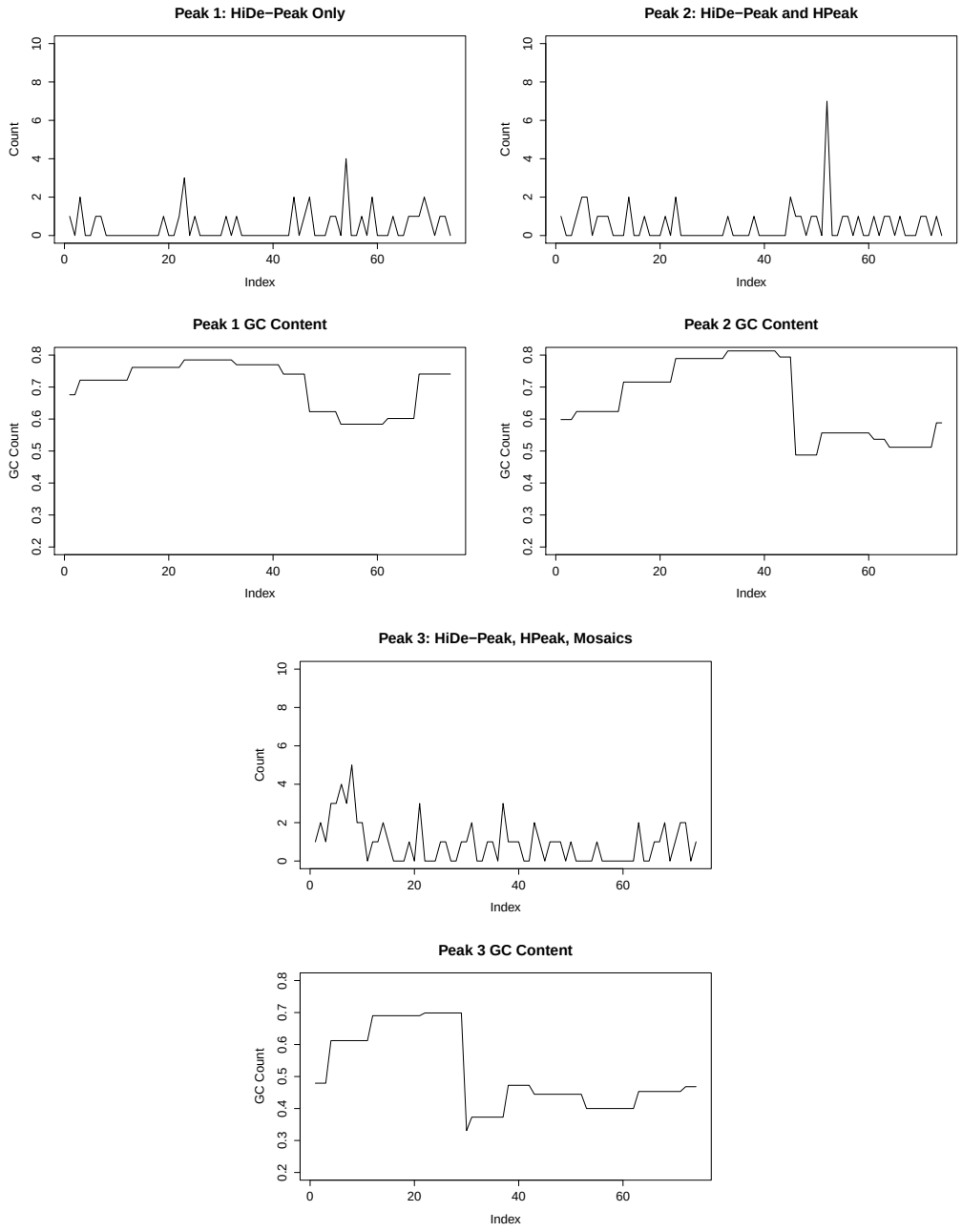


Figure 5.18: Example Peaks from Chromosome 1

CHAPTER 6

# CONCLUSIONS

ChIP-seq data has only been available since 2006, however, a large number of methods have been developed in that time. Early methods, which relied on using a simple cut-off, have been shown to be ineffective. The importance of accounting for the control data has been shown with a variety of new methods. Unfortunately, there is no gold standard, and no way to tell which method performs best. There is no data set in which the true peaks are known. Furthermore, the real data has many unique characteristics that make simulating realistic data difficult.

While accounting for the control is known to improve the results, other factors have not been shown to be as important. Methods that require peaks appearing on both the forward and reverse strands appear to pick up far fewer false positives than other methods, but they do so at the cost of many true positives are can only be identified on one strand. Accounting for covariates can improve results, but only if the covariate signal is particularly strong. Accounting for dependence does appear to make a difference, as HPeak, with its hidden Markov model, is particularly accurate in its peak identification. Furthermore, using a distribution that allows for over dispersion, instead of the usual Poisson, also appear to make a difference.

There is no perfect method to identify peaks. Part of this is due to the experimental process. DNA tags of 25 base pairs are saved, despite the fact that the original read is some unknown longer length of as many as 250 base pairs. Only one replicate of the experiment is run - or more accurately, they combine across all 6 replicates (the lanes), removing additional information. The data set is very large, which makes complex methods more difficult to run, leading to binning and other shortcuts.

The method presented in this thesis attempts to account for the known issues in the data. It accounts for covariates, the read dependence, the strandedness, and the over-dispersion of read counts. But it only barely outperforms other methods in simulation. In addition, these methods present biologists with thousands of peaks to investigate. The small differences in peak finding among the best of these methods may not matter much to biologists with limited time to investigate the results.

## 6.1 Future Work

There remains several things that might be done to improve this method and to prepare the results for publications.

**Lane Effect** If it is possible to get data with the lane information present, it would be worthwhile to model the lane counts separately, and not summed together.

**Coding** Currently, the methods described in this thesis are run with a few separate pieces of code using both Python and R. In order for this to be usable by others, these pieces need to be combined into one easily usable method.

**Motif Analysis** Instead of simulations, most of the published ChIP-seq papers check the accuracy of their peaks with Motif analysis on real data sets. Although motif analysis cannot confirm if the peaks found are 'correct', it can indicate that the peaks are consistent and likely to be true peaks.

**Markov Chain with multiple levels** There has been some interest expressed in developing a method that can identify not only strong peaks, but also weak peaks and background. Adding extra levels to the hidden Markov chain should be straight forward.

APPENDIX A

# CODE

The Viterbi algorithm is a recurring algorithm to help calculate the sequence of 0s and 1s for the hidden Markov chain that maximizes the likelihood for a given set of parameters. The function below calculates the terms of the Viterbi algorithm for HiDe-peak with covariate information.

```
def viterbi(x, lambda_0, lambda_1, beta_0, beta_1, alpha, GC, Map, control, vold, LIM):
    v = range(4)
    if Map == 0:
        p1 = 0
        p0 = 1
    else:
        if GC<.2 or GC>.8
            GC = 0
        a_1 = beta_1[0] + beta_1[1]*GC+beta_1[2]*GC*GC+beta_1[3]*math.sqrt(control)
        mu_1 = math.exp(a_1)
        p1 = dnegbi(x, mu_1, alpha)
        a_0 = beta_0[0]+beta_0[1]*math.sqrt(control)
        mu_0 = math.exp(a_0)
        p0 = dpois(x, mu_0)
    p0 = math.log(p0)
    p1 = math.log(p1)
    m1 = max(math.log(lambda_0)+vold[0], math.log(1-lambda_1)+vold[1])
    v[0] = p0 + m1
    m2 = max(math.log(1-lambda_0)+vold[0], math.log(lambda_1)+vold[1])
    v[1] = p1 + m2
    if(math.log(lambda_0)+vold[0])>(math.log(1-lambda_1)+vold[1]):
        v[2]=0
    else:
        v[2]=1
    if(math.log(1-lambda_0)+vold[0])>(math.log(lambda_1)+vold[1]):
        v[3]=0
    else:
        v[3]=1
    return v
```

Once the Viterbi algorithm has been run for each position, a simple trace-back algorithm is performed to identify the sequence of highest likelihood.

The second part of the estimating procedure is the IRLS algorithm. For the negative binomial peak positions, the first and second derivatives need to be calculated. The code below assumes that the control, GC value, mappability value, and count for each position has been read in from the appropriate files.

```
ln = Regend-Regstart+1 #ln = length of the peak region
Regpos = range(ln)
Regpos= Regpos + Regstart #Regpos is the positions of the peak region
for k in Regpos:
    countcur = count[i]
    controlcur = countcontrol[i]
    if GC[i]>.2 or valGC[i]<.8:
       valGCcur = valGC[i]
    else:
       valGCcur = 0
    if valMap[i] == 0:
        a_i = 0
        mu_i = 0
        loss = 0
        deriv = 0
    else:
        a_i = beta_1[0]+beta_1[1]*valGCcur + beta_1[2]*valGCcur*valGCcur
            + beta_1[3]*math.sqrt(controlcur)
        mu_i = math.exp(a_i)
        loss = countcur - (countcur+alpha)/(mu_i+alpha)*mu_i
        deriv = (countcur+alpha)/(mu_i+alpha)*mu_i*(1-mu_i/(mu_i+alpha))
        p1 = p1 + loss*loss/mu_i
        dl1[0] = dl1[0] + loss
        dl1[1] = dl1[1] + valGCcur*loss
        dl1[2] = dl1[2] + valGCcur*valGCcur*loss
        dl1[3] = dl1[3] + math.sqrt(controlcur)*loss
        d2l1[0] = d2l1[0] + deriv
        d2l1[1] = d2l1[1] + valGCcur*deriv
        d2l1[2] = d2l1[2] + valGCcur*valGCcur*deriv
        d2l1[3] = d2l1[3] + math.sqrt(controlcur)*deriv
        d2l1[4] = d2l1[4] + valGCcur*valGCcur*deriv
        d2l1[5] = d2l1[5] + valGCcur*valGCcur*valGCcur*deriv
        d2l1[6] = d2l1[6] + valGCcur*math.sqrt(controlcur)*deriv
        d2l1[7] = d2l1[7] + valGCcur*valGCcur*valGCcur*valGCcur*deriv
        d2l1[8] = d2l1[8] + valGCcur*valGCcur*math.sqrt(controlcur)*deriv
        d2l1[9] = d2l1[9] + controlcur*deriv
    i = i+1
```

## BIBLIOGRAPHY


[1] Yoshua Bengio and Paolo Frasconi. An input output hmm architecture. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 427–434. MIT Press, 1995.

[2] L Cerchietti K Chatzi E Park L Klemm Y Kim S Herzog H Jumaa M Braig M Kahn A Melnick M Muschen C Hurtz, C Duy. Bcl6 is required for leukemia initiation in chronic myeloid leukemia. *Journal of Experimental Medicine*, 208(11).

[3] Eugenia G Giannopoulou and Olivier Elemento. An integrated chip-seq analysis platform with customizable workflows. *BMC Bioinformatics*, 12(277), 2011.

[4] Raymond K Auerbach Zhengdong D Zhang Theodore Gibson Robert Bjornson Nicholas Carriero Michael Snyder Joel Rozowsky, Ghia Euskirchen and Mark B Gerstein. Peak-seq enables systematic scoring of chip-seq experiments relative to controls. *Nature Biotechnology*, 27:66–75, 2009.

[5] David S. Johnson, Ali Mortazavi, Richard M. Myers, and Barbara Wold. Genome-wide mapping of in vivo protein-dna interactions. *Science*, 316(5830):1497–1502, 2007.

[6] R Jothi, S Cuddapah, A Barski, K Cui, and K Zhao. Genome-wide identification of in vivo protein-dna binding sites from chip-seq data. *Nucleic Acids Res*, 36o:5221–5231, 2008.

[7] Peter Kharchenko, Michael Tostsorukov, and Peter J. Park. Design and analysis of chip-seq experiments for dna-binding proteins. *Nature Biotechnology*, 26(12):1351–1359, 2008.

[8] R. Leadbetter, G. Lindgren, and H. Rootzén. *Extremes and related properties of random sequences and processes*. Springer series in statistics. Springer-Verlag, 1983.

[9] Tarjei S. Mikkelsen, Manching Ku, David B. Jaffe, Biju Issac, Erez Lieberman, Georgia Giannoukos, Pablo Alvarez, William Brockman, Tae-Kyung Kim, Richard P. Koche, William Lee, Eric Mendendhall, Aisling O'Donovan, Aviva Presser, Carsten Russ, Xiaohui Xie, Alexander Meissner, Marius Wernig, Rudolf Jaenisch, Chad Nusbaum, Eric S. Lander, and Bradley E. Bernstein. Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature*, 448:553–561, 2007.

[10] David Nix, Samir Courdy, and Kenneth Boucher. Empirical methods for controlling false

positives and estimating confidence in chip-seq peaks. *BMC Bioinformatics*, 9(1):523, 2008.

[11] Guangjin Pan James A. Thomson Ron Stewart Sndz Keles Pei Fen Kuan, Dongjun Chung. A statistical framework for the analysis of chip-seq data. *Journal of the American Statistical Association*, 106(495):891–903, 2011.

[12] Zhaohui Qin, Jianjun Yu, Jincheng Shen, Christopher Maher, Ming Hu, Shanker Kalyana-Sundaram, Jindan Yu, and Arul Chinnaiyan. Hpeak: an hmm-based algorithm for defining read-enriched regions in chip-seq data. *BMC Bioinformatics*, 11(1):369, 2010.

[13] G Robertson, M Hirst, M Bainbridge, M Bilenky, Y Zhao, T Zeng, G Euskirchen, B Bernier, R Varhol, and A Delaney. Genome-wide profiles of stat1 dna association using chromatin immunoprecipitation and massively parallel sequencing. *Nat Methods*, 4:651–657, 2007.

[14] A Valouev, DS Johnson, A Sundquist, C Medina, E Anton, S Batzoglou, RM Myers, and A Sidow. Genome-wide analysis of transcription factor binding sites based on chip-seq data. *Nature Methods*, 2008.

[15] Clifford A Meyer Jerome Eeckhoute David S Johnson Bradley E Bernstein Chad Sunbaum Rchard M Myers Myles Brown Wei Li Yong Zhang, Tao Liu and X SHirley Liu. Model-based analysis of chip-seq (macs). *Genome Biology*, 9(9), 2008.

[16] Krzywinski M Ning K Droit A Jones S Gottardo R. Zhang X, Robertson G. Pics: Probabilistic inference for chip-seq. *Biometrics*, 67(1), 2011.